Introduction to Gravitational Clustering

Armen Aghajanyan

***Abstract*—The downfall of many supervised learning algorithms, such as neural networks, is the inherent need for a large amount of training data (Benediktsson et al., 1993). Although there is a lot of buzz about big data, there is still the problem of doing classification from a small data-set. Other methods such as support vector machines, although capable of dealing with few samples, are inherently binary classifiers (Cortes and Vapnik, 1995), and are in need of learning strategies such as One vs All in the case of multi-classification. In the presence of a large number of classes this can become problematic. In this paper we present, a novel approach to supervised learning through the method of clustering. Unlike traditional methods such as K-Means (MacQueen, 1967), Gravitational Clustering does not require the initial number of clusters, and automatically builds the clusters, individual samples can be arbitrarily weighted and it requires only few samples while staying resilient to over-fitting.**

***Keywords*—Machine Learning, Classification, Clustering.**

I. Introduction

The name of this algorithm is derived from the metaphor that the algorithm was built upon. Each cluster is symbolic of a planet,and each planet has a mass and a radius as well as the class that it represents. But unlike real life planets, our planets are static with respect to other planets. The process of training can be conceptually thought of as building a universe. The process of predicting is simply placing a mass in the universe and tracing what planet it will appear on.

This algorithm exhibits three nice properties:

1) Ability to learn from a few samples.
2) Ability to weight the importance of training vectors.
3) The nature of the algorithm makes it resilient to overfitting.

The ability to weight the importance of training vectors as well as the ability to learn from a few samples allows us to model a system that supports the notion of prototypes, e.g. Eleanor Rosch (Lakoff, 1987)(P. 41).

II. Definition

Let us start by mathematically defining what each one of our symbolic structures will be. The most important structure is our cluster or our planet. We will define the planet as containing a dynamic mass m, dynamic radius r, dynamic position $\overrightarrow{x}$ and a static class θ. Mathematically:

$$\begin{aligned} m &\in \mathbb{R} \\ r &\in \mathbb{R} \\ \overrightarrow{x} &\in \mathbb{R}^n \\ \theta &\in \mathbb{Z} \end{aligned} \tag{1}$$

$$\mathbb{P} = \{m, r, \overrightarrow{x}, \theta\}$$

Our universe will simply consist of a set of planets. The universe will also hold a couple of global constants. The initial radius of a planet that has just been created which we will denote with r'. The so called percent step, which represents the amount a test mass moves before recalculating the new forces on the test mass. We will denote this with the Greek α. The amount of steps taken or iterations will be denoted with β. The distance between planets will be calculated with the function denoted $D(\overrightarrow{x}, \overrightarrow{y})$.

III. Training Model

One of the better aspects of the model is its ability to rate your feature vectors. To do so, let us define a hybrid feature vector h.

$$h = \{\overrightarrow{x}, m, \theta\} \tag{2}$$

The m variable allows us to rate the value of the feature vector. For example if you have a probabilistic diagnosis, each feature vector will contain the class of the diagnosis as well as the probability of the diagnosis represented by the mass. The training is quite simple. Below is the pseudocode.

nearplanets ← Find Planets in Radius of $\overrightarrow{h}_x$;
nearplanets ← nearplanets where $P_\theta = h_\theta$;
if *nearplanets is Empty* **then**
  Universe Add Planet
  $\{m = h_m, r = r', \overrightarrow{x} = \overrightarrow{h}_x, \theta = h_\theta\}$
else
  p ← planet that generates most force ∈
  nearplanets ;
  Universe update p ←
  $\{m = p_m + h_m, r = m\frac{p_r}{p_m}, \overrightarrow{x} = \frac{p_m}{m}\overrightarrow{p}_x + \frac{h_m}{m}\overrightarrow{h}_x\}$
end

Algorithm 1: Training Algorithm

The new position is a weighted sum of the two position vectors with respect to their weight.

A. Asymptotic Analysis

Our training simply traverses through all of the planets in the universe and computes the distance from the training sample. Saying N is the amount of planets and D is the dimensionality of our feature vectors. Assuming that the planet exists, we get

$$\mathrm{O}\left(D * N\right) = \mathrm{O}\left(N\right) \tag{3}$$

Using a KD-Tree (Bentley, 1975) will allow us to train with the average asymptotic of

$$\mathrm{O}\left(D * \log N\right) = \mathrm{O}\left(\log N\right) \tag{4}$$

On the flip side, assuming we have to add the planet:

$$\mathrm{O}\left(D * N + N_{near}\right) = \mathrm{O}\left(N + N_{near}\right) \tag{5}$$

KD-Tree (Bentley, 1975)

$$\mathrm{O}\left(D * \log N + N_{near}\right) = \mathrm{O}\left(\log N + N_{near}\right) \tag{6}$$

This is the asymptotic of adding a single train vector. Stating that N_s is the number of samples we end up with the final equation being.

$$\mathrm{O}\left(N_s(\log N + N_n)\right) \tag{7}$$

B. Comparison of Training Times

	Gravitational Clustering	K-Means	SVM	Decision Trees
Big O	$\mathrm{O}\left(N_s(\log N + N_n)\right)$	$O(n^{Dk+1}\log n)$	$O(n^3)$	$O(n_s D \log(n_s))$
Online Training	Yes	Yes	No	Partial
Variant Importance	Yes	No	No	No

- N_n is synonymous with N_{near}
- N_s is synonymous with $N_{samples}$

IV. Simulation Testing Model

Metaphorically, predicting the class of a new point is equivalent to dropping a piece of mass into the universe and tracing the mass until it collides with a planet. In this metaphor, we assume that the planets are infinitely small and therefore there will be no interference. Our test point will simply be defined as $l = \{\overrightarrow{x}\}$ Let us first define getting the normalized directional force vector. Recall from physics that the gravitational force between two planets is

$$F = \mathbb{G}\frac{m_1 m_2}{r^2} \tag{8}$$

In our case, we will assume that the mass of each test point is equal to every other, therefore we can disregard the mass. We can also remove the $\mathbb{G}$ constant. Our hybrid force equation per planet p is now:

$$F = \frac{p_m}{r^2} \tag{9}$$

Where r is $D(\overrightarrow{p}_x, \overrightarrow{l}_x)$. We define the total normalized force on our test mass with the custom equation.

$$\begin{aligned} F_{net} &= \textstyle\sum_{p \in Universe} \frac{p_m * (\overrightarrow{p}_x - \overrightarrow{l}_x)}{r^2} \\ F_{norm} &= \frac{\alpha}{||F_{net}||} F_{net} \end{aligned} \tag{10}$$

To restate, α is the percent step taken with respect to the force. Now let us describe the simulation algorithm:

pos $\leftarrow \overrightarrow{l}_x$;
for *i in i* $[0, \beta]$ *step 1* **do**
  force $\leftarrow \sum_{p \in Universe} \frac{p_m * (\overrightarrow{p}_x - pos)}{r^2}$;
  ;
  norm $\leftarrow \frac{\alpha}{||force||} force$;
  pos $\leftarrow$ pos + norm
end
nearplanets $\leftarrow$ Find Planets in Radius of pos;
if *nearplanets is not Empty* **then**
  return **mode**[nearplanets θ]
else
  return [planet closest to pos] θ
end

A. Asymptotic Analysis of Simulation Testing Model

Let us state that N is the number of planets and D is the dimensionality of our feature vector. Calculating the force takes up

$$\mathrm{O}\left(4D * N\right) = \mathrm{O}\left(N\right) \tag{11}$$

The 4 comes from the vector arithmetic that needed to be done. One subtraction, one multiplication, one distance squared, one division. The N term came from the summation. The total simulation next becomes.

$$\mathrm{O}\left(7D * N * \beta + N\right) \tag{12}$$

The 3 more D terms come from: finding the magnitude, multiplying by force (simultaneously multiplying by α) and the update summation. The next N came from finding the planets with the radius containing pos. We can disregard the final if statement since they do not directly affect N. We get:

$$\mathrm{O}\left(7D * N * \beta + N\right) = \mathrm{O}\left(N(7D * \beta + 1)\right) = \mathrm{O}\left(N\right) \tag{13}$$

V. Probabilistic Non-Simulating Model

We propose an different method of computing the class of the test point, without the need of simulation and through purely statistical methods. We first make an assumption that a planet or cluster is normally distributed from the center and the standard deviation is some function of the radius of the planet $\sigma(p_r)$. Therefore let us the define the probability density function.

$$\mathbf{PDF_p} = \frac{1}{2\pi * \sigma(p_r)} e^{\frac{D(\overrightarrow{p}_x, \overrightarrow{l}_x)^2}{2\sigma(p_r)^2}} \tag{14}$$

Now to define our prediction equation:

$$\mathbf{MAX}_\theta[\prod_{p \mid p_\theta=\theta_n}^{Universe} \frac{1}{2\pi * \sigma(p_r)} e^{\frac{D(\vec{p}_x, \vec{l}_x)^2}{2\sigma(p_r)^2}}] \quad (15)$$

To account for the fact that different classes have different amounts of planets, we will transform this function into:

$$\mathbf{MAX}_\theta[\frac{\log \prod_{p \mid p_\theta=\theta_n}^{Universe} e^{(\frac{D(\vec{p}_x, \vec{l}_x)^2}{p_m 2\sigma(p_r)^2})}]}{|p \mid p_\theta = \theta_n|} \quad (16)$$

We removed the normalization constant, due to the fact that this is a relative measure. The bottom of the fraction is the number of planets per class which insures that their is no bias due to the different amounts of clusters with varying radius's. The mass term is added to insure that greater planets have a greater impact on the rating.

Through trial and error we found the best function for $\sigma(p_r)$ was simply p_r^2.

The asymptotic will simply be

$$\mathrm{O}(DN) = \mathrm{O}(N) \quad (17)$$

VI. Testing Results

We tested the algorithm out on the Wisconsin breast cancer data-set (Wolberg and Mangasarian, 1990) (Lichman, 2013). Below are the results.

Gravitational Clustering	$r' = 50$ $\alpha = 0.01$ $\beta = 100$	$r' = 5000$ $\alpha = 0.001$ $\beta = 1000$
Simulated Model	89.65%	90.59%
Probabilistic Model	92.78%	72.41%

It is interesting to note that the larger the clusters and smaller the amount of clusters the less accurate the probabilistic model will be. Unless of course the clusters perfectly model the data that they encapsulate.
We continued our testing by comparing the outputs of some popular out of the box methods. All the other algorithms were implemented in the scikit-learn library (Pedregosa et al., 2011). The data-sets we used were the popular Iris data-set (Lichman, 2013), digits data-set(Pedregosa et al., 2011), Ollivetti data-set (Bevilacqua et al., 2006).

		Algorithm			
Data-sets	GC Prob	GC Sim	SVM (poly)	SVM (rbf)	Naive Bayes (Gaussian)
Iris	98.41%	96.82%	94.66%	97.33%	96%
Digits	86.95%	91.04%	98.99%	25.61%	83.85%
Olivetti	65.5%	77.5%	7.5%	8.5%	99.5%

To show that our algorithm can handle very few samples, we tested the following data-sets again, but this time we only used 1 sample per each class as the training data. Below are the results.

	Accuracy Per Data-set	
Algorithm Type	GC Prob	GC Sim
Iris	93.33%	92.00%
Digits	59.96%	58.18%
Olivetti	63.5%	53.75%

VII. Conclusion

In this paper we introduced a novel technique to clustering and supervised learning that can learn from a few samples, while maintaining a low asymptotic run-time and inherently allowing for arbitrary sample weighting. We compared it to current techniques for classification and showed both the strengths of the algorithm as well as the weaknesses. From the test results we can infer that our algorithm acts consisitenly in both low and high dimensional data, as well as staying consistent in a range of multi-class data-sets. All the code written, including the tests and the algorithm itself can be found on TODO:

Thank you for reading.

References

Benediktsson, J. A., SWAIN, P. H., and ERSOY, O. K. (1993). Conjugate-gradient neural networks in classification of multisource and very-high-dimensional remote sensing data. *International Journal of Remote Sensing*, 14(15):2883–2903.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517.

Bevilacqua, V., Mastronardi, G., Pedone, A., Romanazzi, G., and Daleno, D. (2006). Hidden markov models for recognition using artificial neural networks. 4113:126–134.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Lakoff, G. (1987). *Women, Fire, and Dangerous Things*. The University of Chicago Press.

Lichman, M. (2013). UCI machine learning repository.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. pages 281–297.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Wolberg, W. and Mangasarian, O. (1990). Multisurface method of pattern separation for medical diagnosis applied to breast cytology,. pages 9193–9196.