

PSO Applications

Ben Deeks & Lyndon White

September 9, 2014

Summary of PSO

- Particle Swarm Optimisation is an optimisation technique inspired by flocking birds to search a solution space for an optimal solution
- Each particle considers its personal best location and the swarm's best location when adjusting its velocity
- Technique is appropriate for problems that can be expressed as the optimisation of multiple numerical parameters

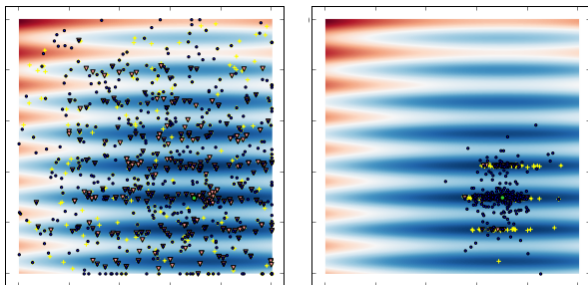


Figure : Before and after shots of PSO at work

PSO for Feature Selection

Pattern Matching and Features

- PSO has application as part of pattern matching and image classification systems
- Data and images can contain a number of "features": components that can be used to distinguish classifications of the data
- Basic image classification algorithms compare differences in the features of images to determine how they are classified
- PSO is useful in **determining** and **weighting** the set of features these algorithms use

Feature Weighting

- Classification algorithms often want to weight features, putting greater weight on features that are better at distinguishing between data
- This can be expressed as an optimisation problem:
Find the set of weightings for features $1..n$ that maximises the accuracy of the classification algorithm
- PSO can be rawly applied as particles in a n dimensional solution space searching for weightings with a range dependent on the implementation and using the accuracy of the classification algorithm as fitness function

The Feature Selection Problem

- It is possible to find a huge number of unintuitive but potentially useful features in data that could be distinguishing
- But it is computationally expensive to look at all the features and suboptimal features could result in loss of accuracy
- We'd like to find a subset X of all features Y with high accuracy, but is no larger than some limit d
- This can be expressed as:

$$J(X) = \max_{Z \subseteq Y, |Z| \leq d} J(Z)$$

Where $J(\cdot)$ is the accuracy of the classification problem

Applicability of PSO

- In the ballpark of PSO as a parameter based optimisation problem
- If the solution is expressed as a sequence of parameters, it would appear as a binary sequence with 0 representing that a feature is not in the set and 1 representing
- For instance

010110

Would represent the subset of features consisting of features 2, 4 and 5

- Looks achievable with a swarm in solution space of dimension n where n is the total number of features with the classifier accuracy function $J(\cdot)$ being the fitness function
- But the parameters are discrete binary values, not continuous values in a space

Discrete Binary PSO

- PSO variation made by Eberhart and Kennedy (1997) for use in discrete binary problem spaces
- Instead of existing in a n -dimensional solution space, particles exist on the vertices of a n -dimensional hypercube
- Particle position is probabilistic. The velocity in a dimension is effectively the probability of being at 1 in that dimension
- So the velocity of a particle must be translated from a value in $(-\infty, +\infty)$ to a probability in $(0, 1)$

Velocity to probability

- To achieve what is needed, change the position function of PSO to:

$$x(t+1) = \begin{cases} 1 & \text{ifrand()} < S(v(t)) \\ 0 & \text{otherwise} \end{cases}$$

- $S(\cdot)$ is the Logistic Function. A function that takes $\mathbb{R} \rightarrow (0, 1)$

$$\frac{1}{1 + e^{-x}}$$

- Given this modification, the PSO algorithm finds an optimal solution in the form of a set of binary parameters

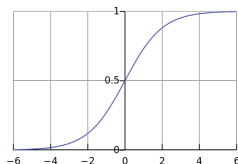


Figure : The Logistic Curve






Solving the problem

- Now we have a PSO algorithm that gives a binary set. Exactly what is needed for the problem
- Established that the PSO will have solution space dimension n , total number of features
- Can use the accuracy function as the fitness function, which will optimise for the best possible set of features
- Need to include a factor that punishes large feature sets into the fitness function. An example fitness function could be:

$$f(X) = \begin{cases} J(X) & \text{if } |X| \leq d \\ 0 & \text{otherwise} \end{cases}$$

PSO for Clustering

Papers

-  A. Ahmadyfard and H. Modares, "Combining pso and k-means to enhance data clustering," in *Telecommunications, 2008. IST 2008. International Symposium on*, Aug 2008, pp. 688–691.
-  C.-Y. Chen and F. Ye, "Particle swarm optimization algorithm and its application to clustering analysis," in *Networking, Sensing and Control, 2004 IEEE International Conference on*, vol. 2. IEEE, 2004, pp. 789–794.
-  F. Ye and C.-Y. Chen, "Alternative kpsso-clustering algorithm," *Tamkang Journal of science and Engineering*, vol. 8, no. 2, p. 165, 2005.
-  C.-Y. Tsai and I.-W. Kao, "Particle swarm optimization with selective particle regeneration for data clustering," *Expert Systems with Applications*, vol. 38, no. 6, pp. 6565–6576, 2011.
-  S. Rana, S. Jasola, and R. Kumar, "A review on particle swarm optimization algorithms and their applications to data clustering," *Artificial Intelligence Review*, vol. 35, no. 3, pp. 211–222, 2011.

Clustering

We have a N -dimensional space, containing data represented as points within it. We will call this space \mathbb{E}_N

To be able to apply a clustering algorithm to any dataset, we first need to map that dataset into points in \mathbb{E}_N . This is a similar notion to the fact that to optimize a problem using PSO we first need to express it as having numerical parameters.

We would like to partition that data into clusters. On the notion that this will give us meaningful groupings.

Clustering Friends

We might for example like to cluster our friends and acquaintances.

We could convert them to points in \mathbb{E}_2 by assigning to each:

- the date when we met
- how many times we have spoken in the past 12 months.

We could expect a clustering algorithm to group all our high school friends in one group, and all our colleges in another – even though it isn't given this information directly.

This is because it is a unsupervised learning algorithm.

What makes a Good Clustering?

- For data from the set X , into clusters $\mathcal{S} = \{C_1, C_2, \dots, C_k\}$
- **A good cluster is compact.** This means it has low variance.
- We can find the centroid of each cluster: $\mu(C) = \frac{1}{|C|} \sum_{\tilde{x} \in C} \tilde{x}$
- So we can find it's variance: $\sigma^2(C) = \sum_{\tilde{x} \in C} (\tilde{x} - \mu(C))^2$
- We would like to minimize the average variance of the clusters.
 - ▶ so minimize: $g_{fit}(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{C \in \mathcal{S}} \sum_{\tilde{x} \in C} (\tilde{x} - \mu(C))^2$

K-Mean Algorithm

Finding the optimum clustering is **NP-Hard**. One of the most common algorithms used to come close is K-Mean.

Begin with a list of Centroids (Z). Then Repeatedly:

- 1 Select clusters (\mathcal{S}) of points closest to those Centroids
- 2 Move of Centroids (Z) to be at center of their cluster

Figure : Source: <http://shabal.in/visuals/kmeans/5.html>

A single step of the K-Mean Algorithm

function K-MEAN-STEP(Z, X)

$S \leftarrow []$

for all $\tilde{x} \in X$ **do**

$\tilde{z} \leftarrow \operatorname{argmin}_{\tilde{z} \in Z} \sqrt{(\tilde{x} - \tilde{z})^2}$

$S[\tilde{z}].append(\tilde{x})$

end for

$Z \leftarrow \{\}$

for all $S \in \mathcal{S}[-]$ **do**

$Z.append\left(\frac{1}{len(S)} \left(\sum_{\tilde{x} \in S} \tilde{x}\right)\right)$

end for

return (\mathcal{S}, Z)

end function

▷ \mathcal{S} is a indexed set of clusters (sets)

▷ For each datum

▷ Find closest centroid

▷ Add it to cluster for that centroid

▷ For each cluster

▷ Calculate new centroid

▷ Return clusters and centroids

K-Mean Algorithm

Initialize Z then:

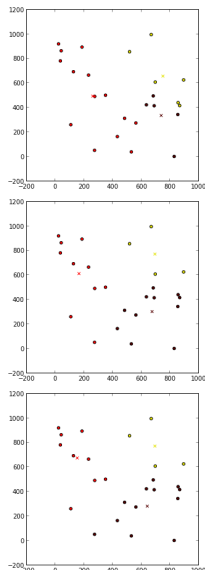
Repeatedly call

$$(\mathcal{S}, Z) \leftarrow \text{K-MEAN-STEP}(Z, X)$$

Until:

- Centroids don't change position (much);
- or: Cluster membership doesn't change;
- or: exceeds preset number of iterations

Figure - Right: K-Mean Iterative execution, which stabilized after 3 generations



K-Mean: Not Globally Optimum

- Once the initial centroids (Z) are set, it is deterministic
- Depending on how initial centroids set get different clusterings of same data
- Not all of which will (necessarily) be equality optimal
- In fact, K-Mean will often get stuck in local optima.

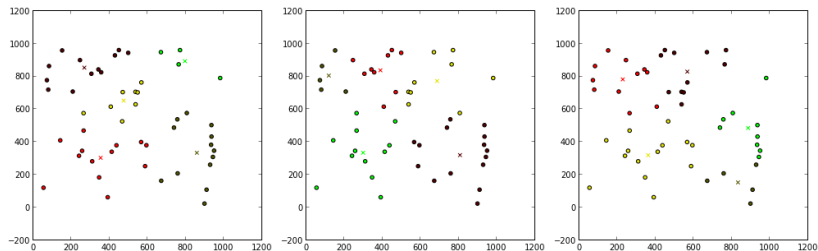


Figure : 3 different Clusterings of same data

Can we express clustering in a PSO accessible form?

- Our goal was find the global optimal clustering
- expressed as minimizing

$$g_{fit}(\mathcal{S} = C_1, C_2, \dots, C_k) = \frac{1}{|\mathcal{S}|} \sum_{\forall C \in \mathcal{S}} \sum_{\forall \tilde{x} \in C} (\tilde{x} - \mu(C))^2$$

- The the centroids can define the clusters, so a mapping exists $\mathcal{C} : Z \rightarrow \mathcal{S} : \tilde{z} \mapsto C_i$ as was done in K-Mean.
- we can re-express g_{fit} as

$$g_{fit}(Z = \{\tilde{z}_1, \tilde{z}_2, \dots, \tilde{z}_k\}) = \frac{1}{|Z|} \sum_{\forall \tilde{z} \in Z} \sum_{\forall \tilde{x} \in \mathcal{C}(\tilde{z})} (\tilde{x} - \tilde{z})^2$$

- This is a function, in terms of numerical parameters Z thus can apply PSO.

What is the fitness function again?

$$g_{fit}(Z = \{\tilde{z}_1, \tilde{z}_2, \dots, \tilde{z}_k\}) = \frac{1}{|Z|} \sum_{\forall \tilde{z} \in Z} \sum_{\forall \tilde{x} \in \mathcal{C}(\tilde{z})} (\tilde{x} - \tilde{z})^2$$

function G-FIT(Z, X)

$\mathcal{C} \leftarrow$ CLUSTER-UP(Z, X)

return $\frac{1}{|Z|} \sum_{\forall \tilde{z} \in Z} \sum_{\forall \tilde{x} \in \mathcal{C}[\tilde{z}]} (\tilde{x} - \tilde{z})^2$

end function

function CLUSTER-UP(Z, X)

$\mathcal{C} \leftarrow []$

for all $\tilde{x} \in X$ **do**

$\tilde{z} \leftarrow \operatorname{argmin}_{\tilde{z} \in Z} \sqrt{(\tilde{x} - \tilde{z})^2}$

$\mathcal{C}[\tilde{z}].\operatorname{append}(\tilde{x})$

end for

return ($\mathcal{C}[_]$)

▷ This may look familiar

PSO for clustering

- Each particle is **not** a centroid
- Each particle is a full set of k centroids.
- that is the particles are flying around the space: $(\mathbb{E}_N)^k$

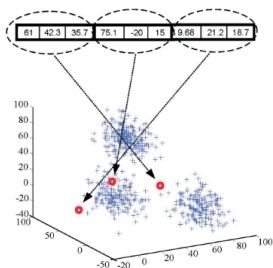


Figure : Position of centroids in \mathbb{E}_3 for a single particle ($K = 3$), from Ye and Chen(2005)

- Chen and Ye (2004) applied the G-Best PSO algorithm to clustering in this way.
- They found it clustered marginally better than K-mean

AKPSO

in 2005 Ye and Chen proposed what they called the Alternative K-Mean PSO algorithm.

Repeat:

- 1 Evaluate fitness function at all particle points
 - ▶ Finding the *pbests* and *gbests*
- 2 Apply a single step of K-Mean to the *gbest* only
 - ▶ moving it to a roughly locally optimum position
- 3 Apply the PSO (GBest) movement function.
 - ▶ using this new improved GBest

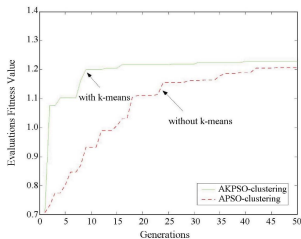


Figure : This worked well (Ye and Chen, 2005))

PSO-KM: K-Mean as final tweaking

- Improving upon Chen and Ye's 2004 work,
 - ▶ and apparently without awareness of their 2005 work,
- Ahmadyfard and Modares (2008), proposed:
 - ① use PSO to train for clustering (as per Chen and Ye, 2004)
 - ★ This gave very good initial values close to (presumed) Global Optima.
 - ② Use full K-Mean to get fine tune the Gbest
 - ★ This is the same notion as applying a hill climber etc, as discussed in the practical lecture.
- This is playing each algorithm to its strengths
 - ▶ PSO as a global search
 - ▶ K-Mean as a local search

K-PSO: starting with some Prior Knowledge

Tsia et. al. (2008), describes K-PSO as:

- 1 execute K-Mean to get set of centroids, use that result as 1 particle.
- 2 Initialise another $5N - 1$ Particles randomly
- 3 Apply a PSO to clustering

K-NM-PSO: Combined Optimising

K-PSO is used by Tsia et. al as stepping stone to the K-NM-PSO: The K-Mean Nelder-Mead Particle Swarm Optimiser

- Initialise $3N$ particles randomly.
- initialise 1 more using K-Mean
- Each iteration:
 - ▶ Apply Nelder-Mead gradient decent to the best N particles
 - ▶ Replace the $N + 1$ th best particle with the result of previous step.
 - ▶ Apply PSO velocity move to worst $2N$ Particles

There are other variations

We have looked at 5 methods for clustering with PSO

- PSO on its own
- PSO with a K-Mean Step on *GBest* once each generation
- PSO, followed by K-Mean fine tuning
- K-mean first to get some prior knowledge, the PSO cluster
- K-NM-PSO for cooperating a lot of optimisers

There are many others: Rana et. al 2008 lists 27 different papers on variations of PSO for Clustering.

End

Questions?