CSCI 4360/6360 Data Science II
Department of Computer Science
University of Georgia

Assignment 3: All-Natural Data Science

Aditya Shinde

September 28, 2017

Partical Swarm Optimization

1

The cognitive and social parameters c_1 and c_2 govern how individual and group decisions contribute to the particle velocity.
The cognitive parameter c_1 is associated with the term $[p_i\vec{(t)} - x_i\vec{(t)}]$. This is the difference between the optimum and the current position of the particle. It will push the particle towards its most optimum position till the current iteration. If this parameter is very high. The particle will keep moving towards the local optimum it has already explored. There will be no exploration just exploitation. If this parameter is very small, the particle will rely very little on its own experience and make decisions based on the group's decisions or based on its previous velocity.
The social parameter c_2 is associated with the term $[g\vec{(t)} - x_i\vec{(t)}]$. This is the difference between the particle's position and the group's optimum position so far. It pushes the particle towards the optimum of the group which may be better or worse than the particle's own optimum. A very high c_2 would mean that the particle would rely heavily on the group's optimum and ignore it's own explored positions which may be better than the group's optimum. A low value for c_2 would mean the opposite. The particle would rely heavily on its own experiences and less on the group's exploration.
If both are small or close to zero, then the particle will just keep moving in a single direction without any knowledge about optima. If both are very high, then their effect will over power the previous velocity. This means that the particles will start moving

towards a single point as soon as they spawn in the search space. This will reduce exploration. r_1 and r_2 further randomise the effect of the social and cognitive infomation on the particle's movement.

2

The parameter ω decides the effect of the particle's previous velocity. During the initial stages, we would want the particles to move to different locations in the space in search for an optima. As the iterations progress, the particle's will have information about which regions have better optima and should start moving towards them. Hence ω should start from 1 and then steadily decrease. During the start $\omega = 1$ will enable to particle to move in the direction it was initialised which is random. And then towards convergence we want the particle to rely more on information about the locations for optima using the social and cognitive terms and so ω should decrease.

3

For particle specific update,
if $f(\vec{x_i}(t)) < \vec{p_i}(t) : \vec{p_i}(t+1) = \vec{x_i}(t)$
else: $\vec{p_i}(t+1) = \vec{p_i}(t)$

For swarm level update:
for i in nparticles:
if $f(\vec{x_i}(t)) < \vec{g}(t) : \vec{g}(t+1) = \vec{x_i}(t)$

4

I would sample r_1 and r_2 from a gaussian distribution with different means instead of a uniform distribution. This way you can have designated explorers and designated particles which move towards optima. So even if we have to trade off training time over convergence, we have some particles which have converged better when other particles are simultaneously searching for better optima.

Linear Dynamical Systems

1

In this case, ignoring the appearance component means we will be fitting the autoregressive model to the data itself rather than its state representation. We can simply assume C as an identity matrix to get $\vec{y_t} = \vec{x_t}$

2

In this problem, $\vec{x_t}$ has dimension R^2. So the matrix X will have dimensions $R^{2\times n}$
From the equation of an autoregressive model,

$$\vec{x_t} = A\vec{x_{t-1}}$$

Applying these equations to the given data,

$$\vec{x_1} = A\vec{x_0}$$

$$\vec{x_2} = A\vec{x_1}$$

$$\vec{x_n} = A\vec{x_{n-1}}$$

$$\vec{x_{n+1}} = A\vec{x_n}$$

So the equation to compute A is,

$$A = X_1^n (X_0^{n-1})^{-1}$$

3

For this problem, I had to read up on eigen decomposition and power iteration method on the web.

Using the power iteration method, if $\vec{v_n}$ is an approximation of the dominant Eigen vector, then

$$\vec{v}_{t+1} = M\vec{v_t}$$

And so, $\vec{v}_{t+2}$ will be written as,

$$\vec{v}_{t+2} = M\vec{v}_{t+1}$$

$$\vec{v}_{t+2} = M^2\vec{v_t}$$

Similarly, $\vec{v}_{t+3}$ will be written as,

$$\vec{v}_{t+3} = M\vec{v}_{t+2}$$

$$\vec{v}_{t+3} = M^3\vec{v_t}$$

For n steps, the equation will be,

$$\vec{v}_{t+n} = M^n\vec{v_t}$$

4

If the eigen decomposition of M is used instead, then the equations generalise to,

$$\vec{v}_{t+n} = \Sigma^n \vec{v}_t$$

Since $U^T U = I$ and $I^n = I$, it will only depend on Σ.
As $n \to \text{inf}$, $\vec{v}$ will converge to the dominant eigen vector.
if $\lambda < 1$ then as the iterations increase, $\vec{v}$ will become smaller. It will still converge to the eigen vector after rescaling.
Similarly, if $\lambda > 1$ then $\vec{v}$ will increase to a huge value.