Self-learning Neural Network Based Traffic Signal Controller for an Isolated Intersection and Construction of a New Clustering Algorithm in Unsupervised Machine Learning

Ravil Mussabayev

May 18, 2018

Problem statement

Build an adaptive traffic signal controller for an isolated intersection. In prospect, it will be extended to a transport network of arbitrary size.

Solution method

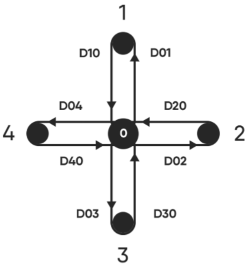
Reinforcement learning has been used to solve the problem.

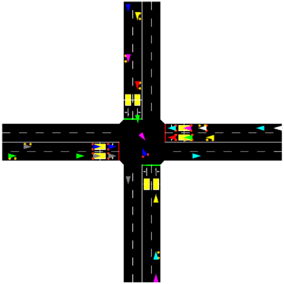
Basic elements of reinforcement learning:

1. The agent and the actions
2. The environment and its states
3. A reward signal

Objective: through successive interaction with the environment train the agent to take actions that maximize the total reward it receives over the long run.

State space





$$NS \doteq D10 + D30$$

$$WE \doteq D40 + D20$$

$$S \doteq (NS, WE)$$

The state space: $\mathcal{S} \doteq \{(s_0, s_1) \mid s_0, s_1 \in \mathbb{Z}^+\}$

Action space

There are 3 actions:

1. $a_0 \doteq (0,0)$ – do nothing
2. $a_1 \doteq (+dt, -dt)$ – extend the NS green phase and shorten the WE green phase
3. $a_2 \doteq (-dt, +dt)$ – shorten the NS greed phase and extend the WE green phase

The action space: $\mathcal{A} \doteq \{a_0, a_1, a_2\}$

Reward formula

The following exlusive continuous reward formula has been used:

$$S_t = (NS, WE) \to S_{t+1} = (NS', WE') \tag{1}$$

$$\begin{aligned} R(S_t, S_{t+1}) \quad & \doteq \mu \overbrace{\left(|NS - WE| - |NS' - WE'|\right)}^{\text{equilibrium term}} + \\ & + (1-\mu) \underbrace{\left(NS + WE - \left(NS' + WE'\right)\right)}_{\text{queue reduction term}} \end{aligned} \tag{2}$$

$\mu \in [0, 1]$ is the trade-off between the queue reduction and equilibrium terms.

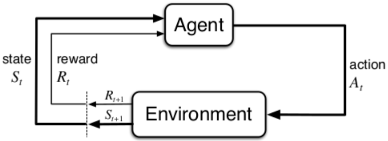
The reward space: $\mathcal{R} \doteq \{R(S_0, S_1) \mid S_0, S_1 \in \mathcal{S}\}$

Markov decision process (MDP)

Discrete time steps: $t = 0, 1, 2, 3, \ldots$
At each time step the agent-environment interaction is:

$$S_t \underset{A_t}{\longrightarrow} (R_{t+1}, S_{t+1}),$$

where $S_t, S_{t+1} \in \mathcal{S}$, $A_t \in \mathcal{A}$, $R_{t+1} \in \mathcal{R}$.



Return and policy

The return:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots, \; 0 \leq \gamma \leq 1 \tag{3}$$

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \\ &= R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \dots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned} \tag{4}$$

The policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$:

$$\pi(a|s) \doteq \mathsf{Pr}\{A_t = a \mid S_t = s\} \tag{5}$$

State value function

Given π, the state value function:

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \end{aligned} \tag{6}$$

State-action value function

Given π, the state-action value function:

$$\begin{aligned} q_\pi(s,a) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \sum_{s',r} p(s', r|s,a)[r + \gamma v_\pi(s')] \end{aligned} \tag{7}$$

The goal

Let $\pi \geq \pi'$ if $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$

$$v_*(s) \doteq \max_\pi v_\pi(s), \ \forall s \in \mathcal{S} \tag{8}$$

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a), \ \forall s \in \mathcal{S} \text{ and } \forall a \in \mathcal{A} \tag{9}$$

Bellman optimality: $v_*(s) = \max_{a \in \mathcal{A}} q_*(s,a)$.

The greedy policy is one of the optimal ones:

$$\pi_*(s) \doteq \operatorname*{argmax}_a q_*(s,a) \tag{10}$$

Shallow Q-Network (SQN)

We approximate the true $q_*(s, a)$ by $\hat{q}(s, a, w)$.

At each time t the cost function is the squared error:

$$J_t(w_t) \doteq \frac{1}{2}\left(q_*(S_t, A_t) - \hat{q}(S_t, A_t, w_t)\right)^2 \tag{11}$$

We don't know $q_*(S_t, A_t)$, so we estimate it:

$$q_*(S_t, A_t) \approx U_t(S_t, A_t) \doteq R_{t+1} + \gamma \max_{a' \in \mathcal{A}} \hat{q}(S_{t+1}, a', w_t) \tag{12}$$

Shallow Q-Network (SQN)

$$J_t(w_t) \doteq \frac{1}{2}\left(U_t(S_t, A_t) - \hat{q}(S_t, A_t, w_t)\right)^2 \tag{13}$$

$$\nabla J_t(w_t) = \left(U_t(S_t, A_t) - \hat{q}(S_t, A_t, w_t)\right) \nabla_w \hat{q}(S_t, A_t, w_t) \tag{14}$$

Update the weights by the gradient descent:

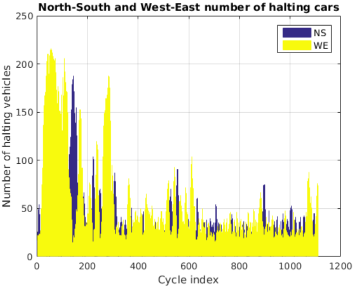
$$w_{t+1} \doteq w_t - \alpha \nabla J_t(w_t) \tag{15}$$

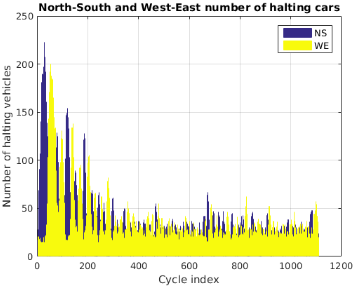
$\nabla J_t(w_t)$ is computed by the backpropagation algorithm.
α is the learning rate.

Shallow Q-Network (SQN). Implementation

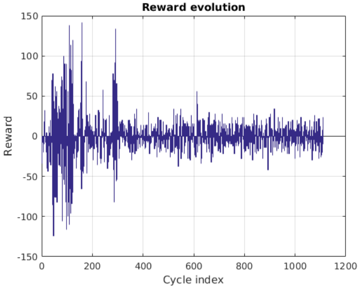
- ε-greedy policy ($\varepsilon = 0.1$): choose action $\mathrm{argmax}_a \, \hat{q}(S_t, a, w_t)$ with probability 0.9, and a random action with probability 0.1
- Discounting factor $\gamma = 0.6$
- No-library Python 2-25-3 NN implementation. ReLU activation function in the hidden layer. Gradient descent with $\alpha = 0.001$, 5 iterations with regularization $\lambda = 0.03$
- Feature scaling and mean normalization ([-1,1])
- 1000 cycles of SUMO simulation

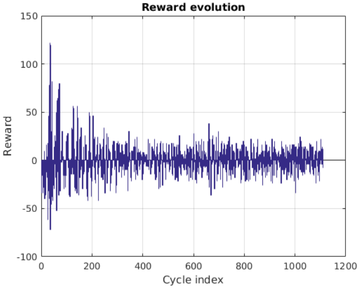
Experiments. SQN (right) vs. Q-learning (left)



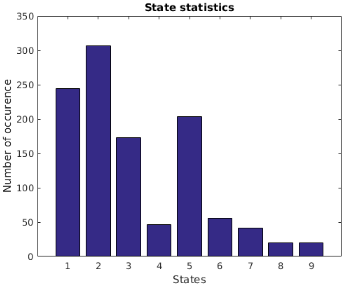


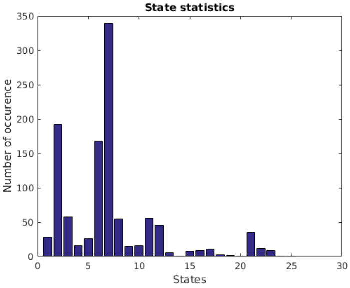
Experiments. SQN (right) vs. Q-learning (left)



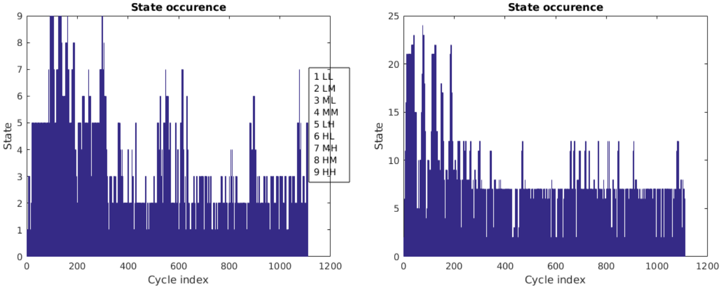


Experiments. SQN (right) vs. Q-learning (left)





Experiments. SQN (right) vs. Q-learning (left)



Why clustering?

Apply an efficient clustering algorithm to the history of occured states to obtain a good discretization of the state space.

Clustering. Problem statement

Let the dataset $D \doteq \{(x_i, y_i)\}_{i=1}^{N}$ be given.

Objective: identify the groups of points that are in some sense similar to each other.

The new clustering algorithm should:

1. be fast enough to be able to work in the online way
2. address the problem of clusters of various densities
3. be able to identify clusters of complex shapes

Local density function

$$f(x,y) \doteq \sum_{(x_i,y_i)\in D} e^{-\lambda((x-x_i)^2+(y-y_i)^2)}$$

$$I = \begin{pmatrix} E & F \\ F & G \end{pmatrix} = \begin{pmatrix} 1+f_x^2 & f_x f_y \\ f_x f_y & 1+f_y^2 \end{pmatrix}$$

$$II = \begin{pmatrix} L & M \\ M & N \end{pmatrix} = \begin{pmatrix} -f_{xx} & -f_{xy} \\ -f_{xy} & -f_{yy} \end{pmatrix}$$

Mean curvature:

$$H = \frac{EN + GL - 2FM}{2(EG - F^2)}$$

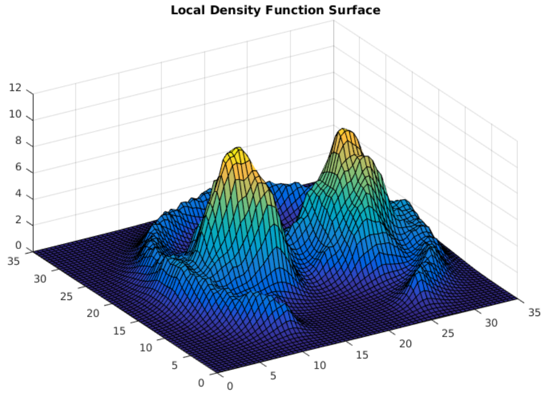
Hypothesis

My conjecture: shift the points along the gradient of the mean curvature in order to obtain a skeleton of each cluster.

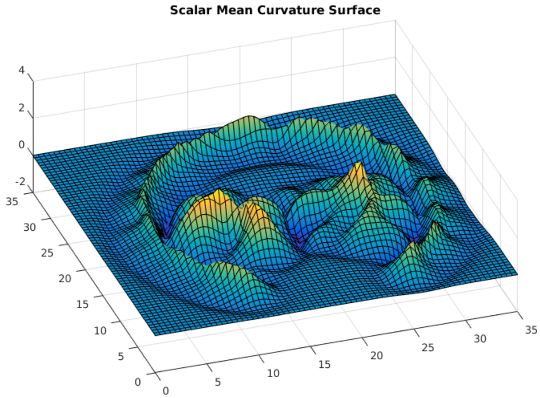
Classification of a new point: classify to the cluster the skeleton of which is nearest to the point.

So far the idea is incomplete, since it is not well understood how to distinguish skeletons of various clusters.

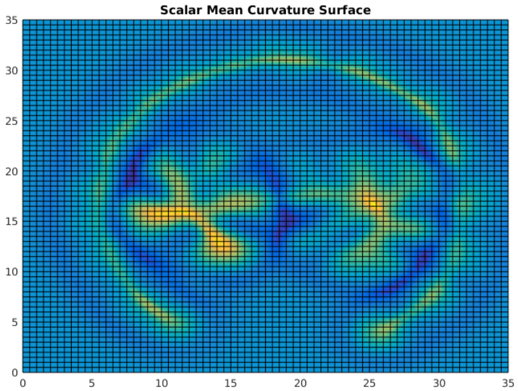
Experiments



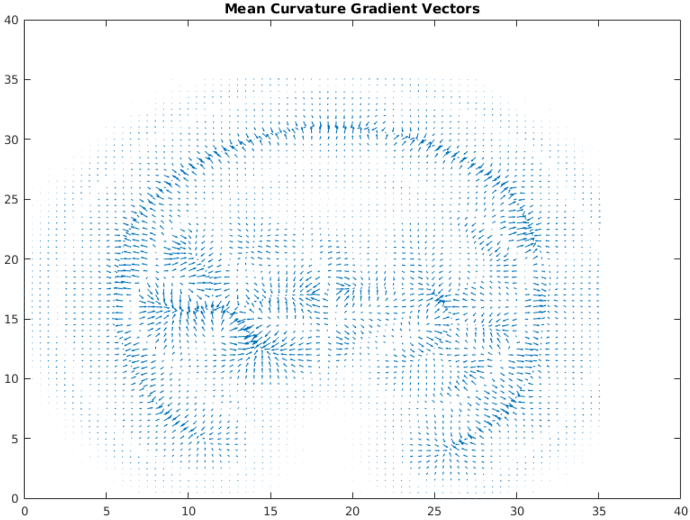
Experiments



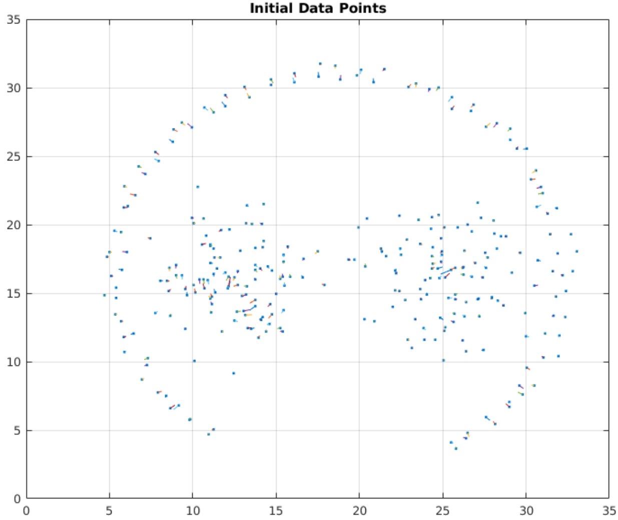
Experiments



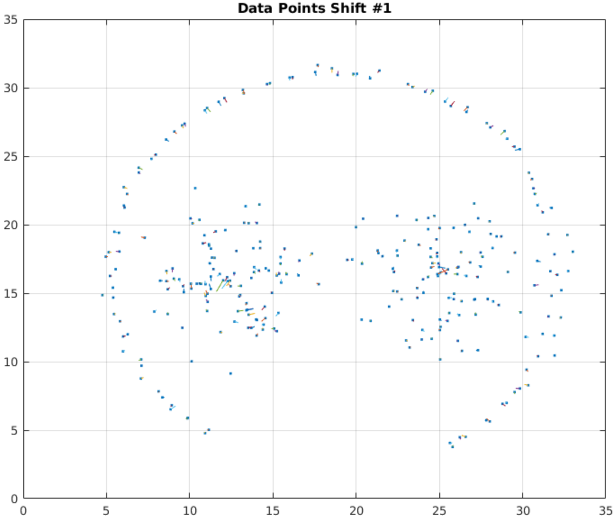
Experiments



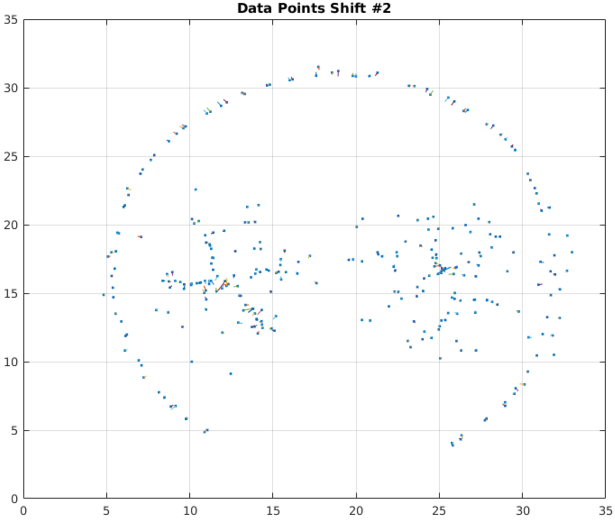
Experiments



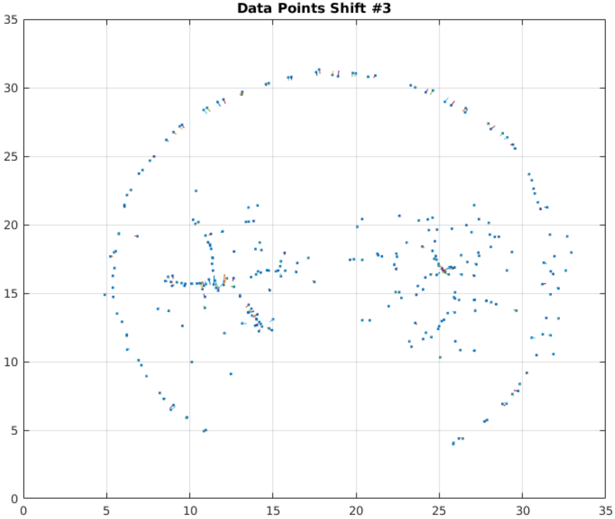
Experiments



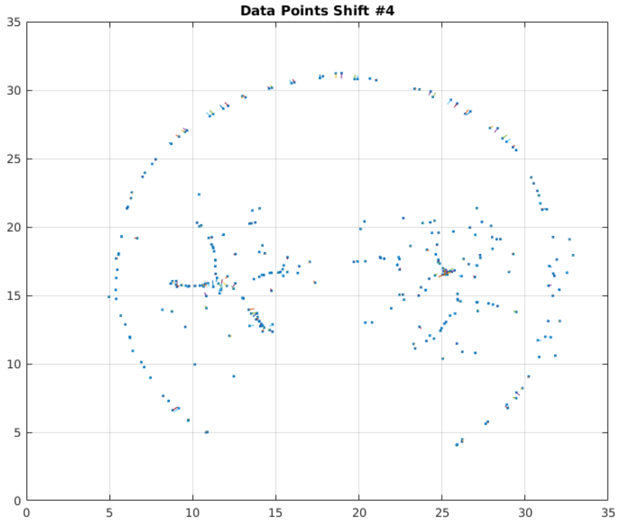
Experiments



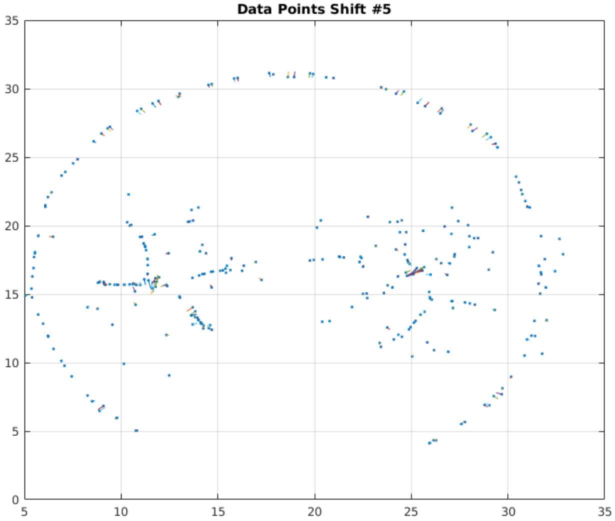
Experiments



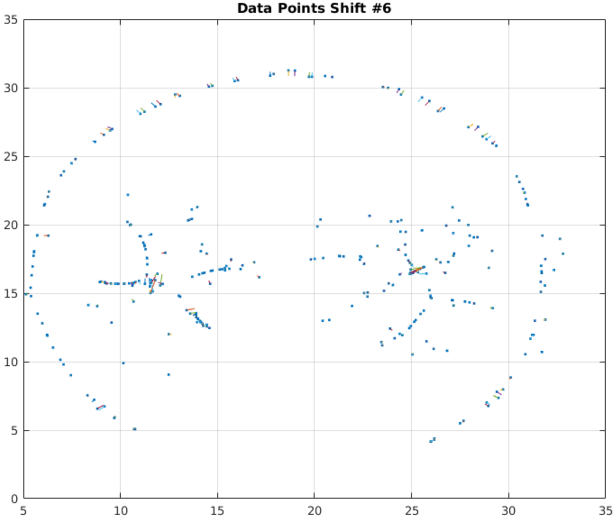
Experiments



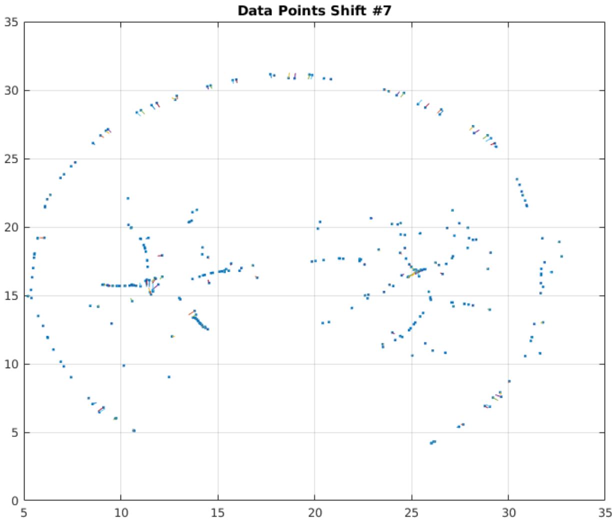
Experiments



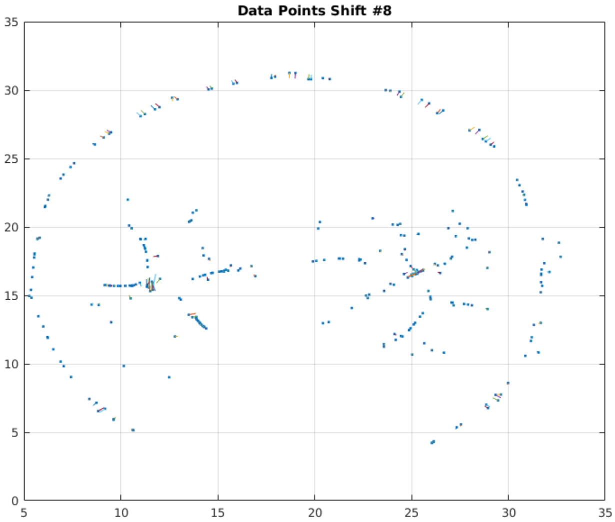
Experiments



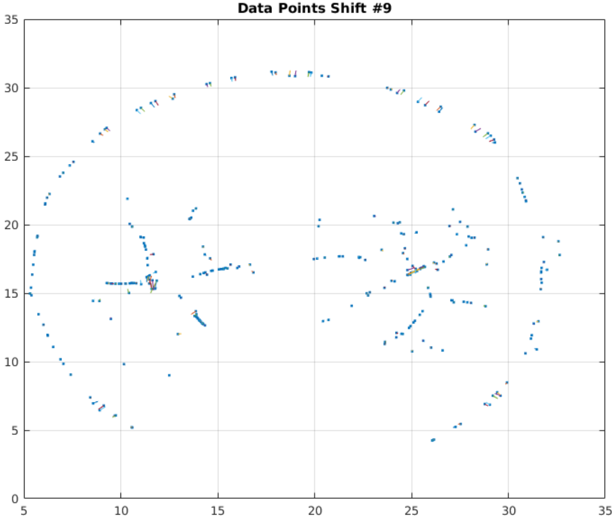
Experiments



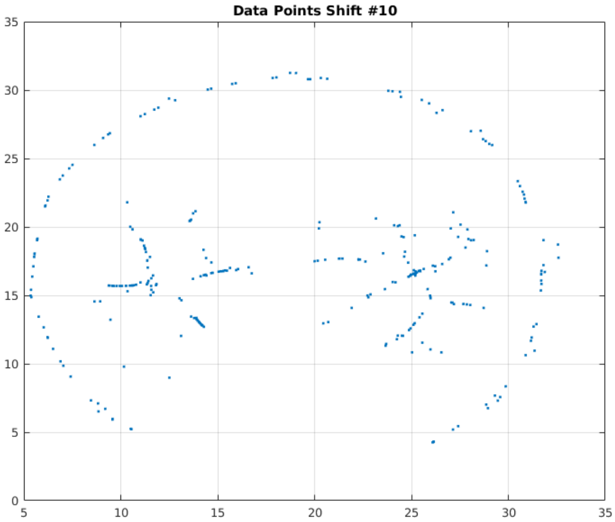
Experiments



Experiments



Experiments



Future work

The SQN algorithm is amenable to the following improvements:

1. We could make the neural network to directly produce the phases duration
2. Include the current signal plan into the input of the neural network, as well as other factors (e.g. time)
3. Apply the algorithm to the transport network consisting of multiple intersections