Reducing Supervisor Burden in DAgger by one class SVM

Huan Ling

2019-01-25

Review: Dagger

(Ross & Gordon & Bagnell, 2011): DAgger, or Dataset Aggregation

- Imitation learning as interactive supervision
- Aggregate training data from expert with test data from execution

Algorithm 1 DAgger

```
D = {(s, a)} initial expert demonstrations
θ_1 ← train learner's policy parameters on D
for i = 1...N do
    Execute learner's policy π_{θ_i}, get visited states S_{θ_i} = {s_0, ..., s_T}
    Query the expert at those states to get actions A = {a_0, ..., a_T}
    Aggregate dataset D = D ∪ {(s, a) | s ∈ S_{θ_i}, a ∈ A}
    Train learner's policy π_{θ_{i+1}} on dataset D
Return one of the policies π_{θ_i} that performs best on validation set
```

Review: Dagger

(Ross & Gordon & Bagnell, 2011): DAgger, or Dataset Aggregation

- Imitation learning as interactive supervision
- Aggregate training data from expert with test data from execution

Algorithm 1 DAgger

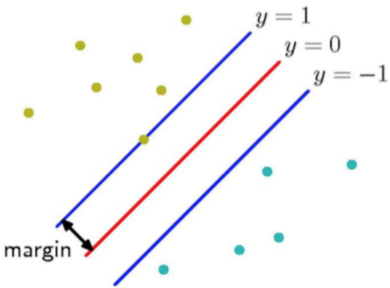
$D = \{(s, a)\}$ initial expert demonstrations
$\theta_1 \leftarrow$ train learner's policy parameters on D
for $i = 1 \ldots N$ **do**
  Execute learner's policy π_{θ_i}, get visited states $S_{\theta_i} = \{s_0, \ldots, s_T\}$
  Query the expert at those states to get actions $A = \{a_0, \ldots, a_T\}$
  Aggregate dataset $D = D \cup \{(s, a) \mid s \in S_{\theta_i},\ a \in A\}$
  Train learner's policy $\pi_{\theta_{i+1}}$ on dataset D
Return one of the policies π_{θ_i} that performs best on validation set

- Dagger is time-consuming and it requires human to annotate data for each state.
- Can we save some human annotations by algorithm?

Two Class SVM Review

- We are given training data $(x^{(i)}, t^{(i)})$.
- We look at classification, so $t^{(i)}$ will represent the class label.
- We use $t = 1$ for the positive and $t = -1$ for the negative class.

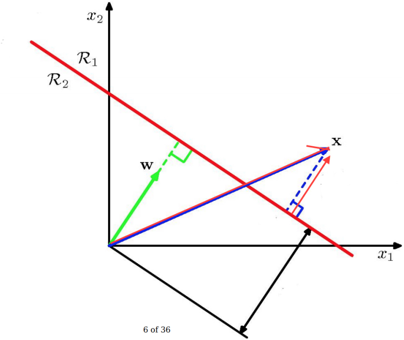
Two Class SVM Review



- **Goal**: Find classification boundary that leads to the largest margin (buffer) from points on both sides.
- Subset of vectors that support (determine boundary) are called the **support vectors**.

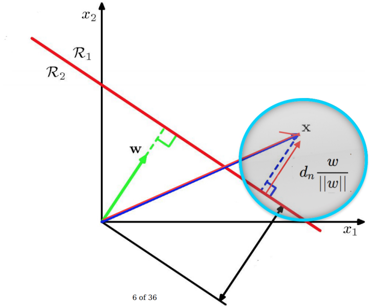
Two Class SVM Review

- Let dicision boundary to be $\boldsymbol{\omega}^{\boldsymbol{T}}\boldsymbol{x} + b = 0$, so that $\boldsymbol{\omega}$ perpendicular to decision boundary.
- Let $\boldsymbol{x}$ to be support vector, d_n is the distance from support vector to boundary.



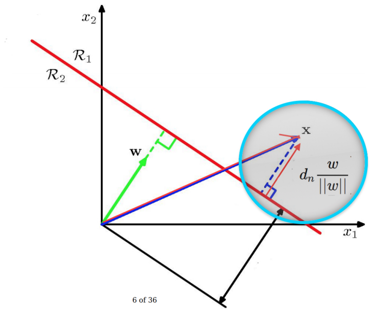
Two Class SVM Review

- $d_n \frac{\omega}{||\omega||}$ is the vector perpendicular to decision boundary and has length d_n.



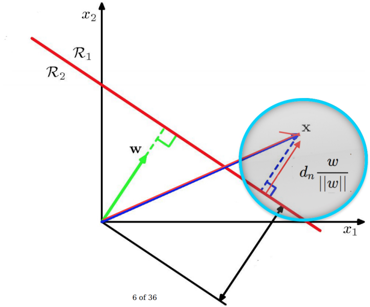
Two Class SVM Review

- $d_n \dfrac{\omega}{||\omega||}$ is the vector perpendicular to decision boundary and has length d_n.
- Vector $x + d_n \dfrac{\omega}{||\omega||}$ locates on decision boundary.



Two Class SVM Review

- $d_n \frac{\omega}{||\omega||}$ is the vector perpendicular to decision boundary and has length d_n.
- Vector $x + d_n \frac{\omega}{||\omega||}$ locates on decision boundary.
- So $\boldsymbol{\omega}^{\boldsymbol{T}}(x + d_n \frac{\omega}{||\omega||}) + b = 0$



Two Class SVM Review

- $d_n \frac{\omega}{||\omega||}$ is the vector perpendicular to decision boundary and has length d_n.

Two Class SVM Review

- $d_n \frac{\omega}{||\omega||}$ is the vector perpendicular to decision boundary and has length d_n.
- Vector $x + d_n \frac{\omega}{||\omega||}$ locates on decision boundary.

Two Class SVM Review

- $d_n \frac{\omega}{||\omega||}$ is the vector perpendicular to decision boundary and has length d_n.
- Vector $x + d_n \frac{\omega}{||\omega||}$ locates on decision boundary.
- So $\boldsymbol{\omega}^{\boldsymbol{T}}(x + d_n \frac{\omega}{||\omega||}) + b = 0$

Two Class SVM Review

- $d_n \frac{\omega}{||\omega||}$ is the vector perpendicular to decision boundary and has length d_n.
- Vector $x + d_n \frac{\omega}{||\omega||}$ locates on decision boundary.
- So $\boldsymbol{\omega}^{\boldsymbol{T}}(x + d_n \frac{\omega}{||\omega||}) + b = 0$
- Derive d_n we have, $d_n = \frac{\omega^T x + b}{||\omega||}$

Two Class SVM Review

- $d_n \frac{\omega}{||\omega||}$ is the vector perpendicular to decision boundary and has length d_n.
- Vector $x + d_n \frac{\omega}{||\omega||}$ locates on decision boundary.
- So $\boldsymbol{\omega}^{\boldsymbol{T}}(x + d_n \frac{\omega}{||\omega||}) + b = 0$
- Derive d_n we have, $d_n = \dfrac{\omega^T x + b}{||\omega||}$
- We let $t_n \in \{-1, 1\}$ stands for label for current data x, then $d_n = \dfrac{t_n(\omega^T x + b)}{||\omega||}$

Two Class SVM Review

We can set $d_n = \frac{1}{||w||}$ for the point x_n closest to the decision boundary, leading to the problem:

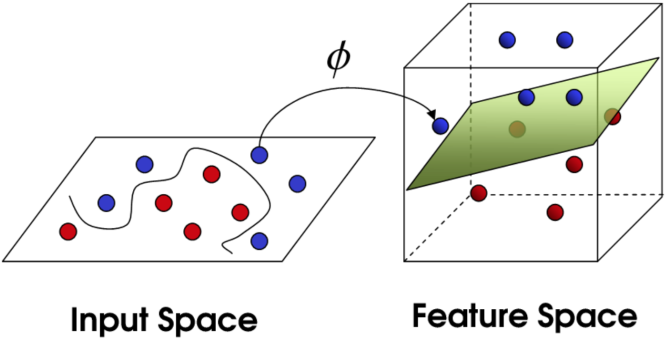
$$\max \frac{1}{||w||}$$
$$\text{s.t. } t_n(w^T x_n + b) \geq 1, \text{ for } n = 1 \dots N$$

- Or equivalently:

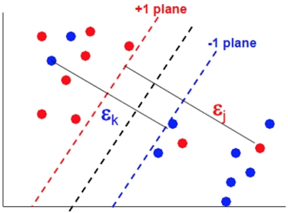
$$\min \frac{1}{2}||w||^2$$
$$\text{s.t. } t_n(w^T x_n + b) \geq 1, \text{ for } n = 1 \dots N$$

Non-linear SVM

- We change $y(x) = \boldsymbol{\omega}^{\boldsymbol{T}}(x) + b$ to $y(x) = \boldsymbol{\omega}^{\boldsymbol{T}}\phi((x)) + b$
- $\phi()$ is called kernel function.



Two Class SVM Review, slack variables



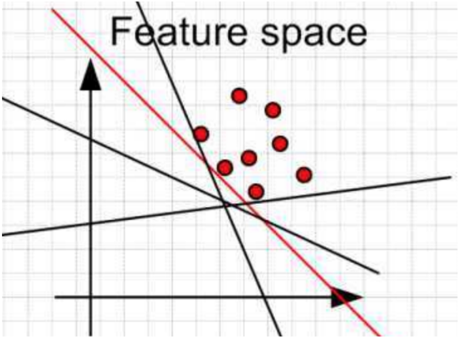
- Introduce slack variables ξ_i

$$\min \frac{1}{2}||\mathbf{w}||^2 + \lambda \sum_{i=1}^{N} \xi_i$$

$$\text{s.t} \quad \xi_i \geq 0; \quad \forall i \quad t^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + b) \geq 1 - \xi_i$$

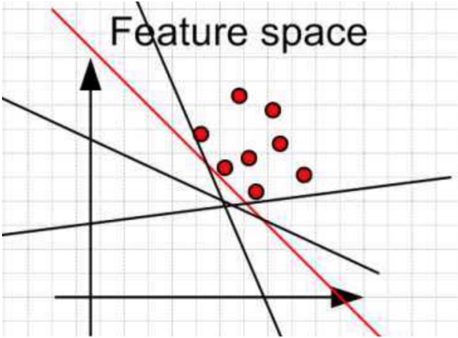
- Example x_i lies on wrong side of hyperplane then $\xi_i > 1$
- So $\sum_i \xi_i$ is the upper bounds of the number of training error.

One Class SVM



- We want to separate all data from origin.
- We want to the decision boundary to be as far as possible to origin.
- The red line above is the ideal decision boundary we want.

One Class SVM



- Why do we want one class SVM?
- One class SVM means all data in training set are in the same class.
- When a new data comes in, we know whether the data is worth labeling.

One Class SVM

$$\begin{aligned}\min_{\boldsymbol{w},b,\boldsymbol{\xi},\rho} \quad & \frac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} - \rho + \frac{1}{\nu l}\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & \boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}_i) - \rho \geq -\xi_i \\ & \xi_i \geq 0,\ i = 1, \cdots, l\end{aligned}$$

- Same as two class SVM, we add slack term v.
- The parameter v controls the penalty or slack term and is an upper bound on the fraction of outliers in traingin set.
- In another word, when $v = 0.1$ then only 0.1 of training data will be claffified to wrong class.

One Class SVM

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi},\rho} \quad \frac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} - \rho + \frac{1}{\nu l}\sum_{i=1}^{l}\xi_i$$

$$\text{subject to} \quad \boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}_i) - \rho \geq -\xi_i$$

$$\xi_i \geq 0, \; i = 1, \cdots, l$$

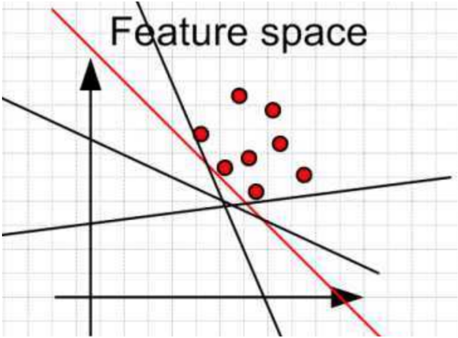
- Same as two class SVM, we add slack term v.
- The parameter v controls the penalty or slack term and is an upper bound on the fraction of outliers in traingin set.
- In another word, when $v = 0.1$ then only 0.1 of training data will be claffified to wrong class.

One Class SVM

$$
\begin{aligned}
\min_{\boldsymbol{w},b,\boldsymbol{\xi},\rho} \quad & \frac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} - \rho + \frac{1}{\nu l}\sum_{i=1}^{l}\xi_i \\
\text{subject to} \quad & \boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}_i) - \rho \geq -\xi_i \\
& \xi_i \geq 0,\ i = 1, \cdots, l
\end{aligned}
$$

- Same as two class SVM, we add slack term v.
- The parameter v controls the penalty or slack term and is an upper bound on the fraction of outliers in traingin set.
- In another word, when $v = 0.1$ then only 0.1 of training data will be claffified to wrong class.

One Class SVM



- One class SVM means all data in training set are in the same class.
- When a new data comes in, we know whether the data is worth labeling.
- However, standard one class SVM doesn't have sense of different classes.

One Class SVM

- We separate our training data in two category, success and failure based on y:

$$y_i = \begin{cases} 1 & : l(\pi_\theta(\mathbf{x}_i), \mathbf{u}_i) \leq \varepsilon \\ -1 & : l(\pi_\theta(\mathbf{x}_i), \mathbf{u}_i) > \varepsilon \end{cases}$$

Where as l is loss function, π_θ is current policy, ε is human defined threshold.

One Class SVM

- We separate our training data in two category, success and failure based on y:

$$y_i = \begin{cases} 1 & : l(\pi_\theta(\mathbf{x}_i), \mathbf{u}_i) \leq \varepsilon \\ -1 & : l(\pi_\theta(\mathbf{x}_i), \mathbf{u}_i) > \varepsilon \end{cases}$$

  Where as l is loss function, π_θ is current policy, ε is human defined threshold.
- $D_s = \{\{x_i, u_i\} \in D_k, y_i = 1\}$, $D_r = \{\{x_i, u_i\} \in D_k, y_i = -1\}$

One Class SVM

- We separate our training data in two category, success and failure based on y:

$$y_i = \begin{cases} 1 & : l(\pi_\theta(\mathbf{x}_i), \mathbf{u}_i) \leq \varepsilon \\ -1 & : l(\pi_\theta(\mathbf{x}_i), \mathbf{u}_i) > \varepsilon \end{cases}$$

  Where as l is loss function, π_θ is current policy, ε is human defined threshold.
- $D_s = \{\{x_i, u_i\} \in D_k, y_i = 1\}$, $D_r = \{\{x_i, u_i\} \in D_k, y_i = -1\}$
- A separate One-Class SVM is then trained on each set of states,and providing measures of both set, g_s and g_r.

One Class SVM

- We separate our training data in two category, success and failure based on y:

$$y_i = \begin{cases} 1 & : l(\pi_\theta(\mathbf{x}_i), \mathbf{u}_i) \leq \varepsilon \\ -1 & : l(\pi_\theta(\mathbf{x}_i), \mathbf{u}_i) > \varepsilon \end{cases}$$

  Where as l is loss function, π_θ is current policy, ε is human defined threshold.

- $D_s = \{\{x_i, u_i\} \in D_k, y_i = 1\}$, $D_r = \{\{x_i, u_i\} \in D_k, y_i = -1\}$
- A separate One-Class SVM is then trained on each set of states,and providing measures of both set, g_s and g_r.
- So our decision function becomes:

$$g_\sigma(\mathbf{x}) = \begin{cases} 0 & : g_s(\mathbf{x}) == 1 \text{ and } g_r(\mathbf{x}) == -1 \\ -1 & : \text{otherwise} \end{cases}$$

SHIV and DAgger

- Both SHIV and DAgger solve the minimization loss iterating two steps.

SHIV and DAgger

- Both SHIV and DAgger solve the minimization loss iterating two steps.
- **Step 1**: Compute a θ using the training data so far

SHIV and DAgger

- Both SHIV and DAgger solve the minimization loss iterating two steps.
- **Step 1**: Compute a θ using the training data so far
- **Step 2**: Execute the policy induced by the current θ, and ask for labels for the encountered states.

SHIV and DAgger

- Both SHIV and DAgger solve the minimization loss iterating two steps.
- **Step 1**: Compute a θ using the training data so far
- **Step 2**: Execute the policy induced by the current θ, and ask for labels for the encountered states.
- **DAgger**: Query supervisor for every new state.

$$D_{k+1} = \mathcal{D}_k \cup \{(\mathbf{x}_t, \tilde{\mathbf{u}}_t) \| t \in \{0, \ldots, T\}\}$$

SHIV and DAgger

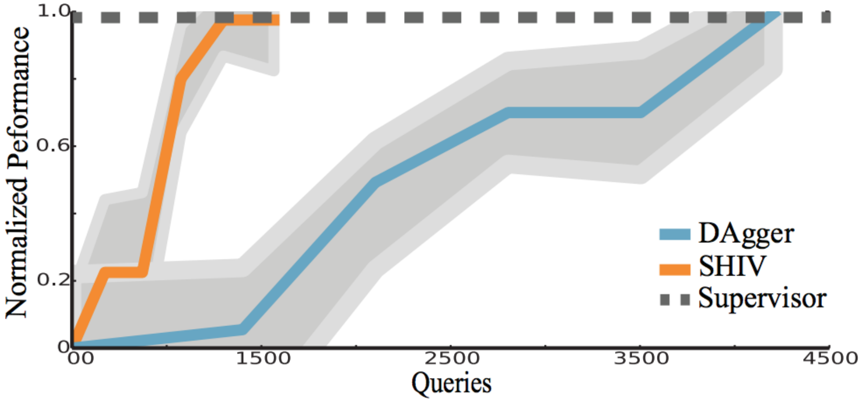
- Both SHIV and DAgger solve the minimization loss iterating two steps.
- **Step 1**: Compute a θ using the training data so far
- **Step 2**: Execute the policy induced by the current θ, and ask for labels for the encountered states.
- **DAgger**: Query supervisor for every new state.

  $$D_{k+1} = \mathcal{D}_k \cup \{(\mathbf{x}_t, \tilde{\mathbf{u}}_t) \| t \in \{0, \ldots, T\}\}$$

- **SHIV**: Using One Class SVM to decide whether to ask a query.

  $$D_{k+1} = \mathcal{D}_k \cup \{(\mathbf{x}_t, \tilde{\mathbf{u}}_t) \| t \in \{0, \ldots, T\}, g(\mathbf{x}_t) = -1\}$$

Results



1.0
0.6
0.2
0
Normalized Peformance
00
1500
2500
3500
4500
Queries
DAgger
SHIV
Supervisor