Implementación y Verificación del Hybrid Normalized Advantage Function (HNAF)

Documentación Técnica

31 de julio de 2025

Índice

1. Introducción

El **Hybrid Normalized Advantage Function (HNAF)** es una extensión del Normalized Advantage Function (NAF) que resuelve problemas de optimización con múltiples modos de funcionamiento. Esta implementación aborda sistemas dinámicos donde la política óptima requiere seleccionar entre diferentes transformaciones según el estado actual.

2. Fundamentos Matemáticos

2.1. Problema Base: Sistema Dinámico Multi-Modo

Consideramos un sistema dinámico discreto con N modos de funcionamiento:

$$\mathbf{x}_{t+1} = f_i(\mathbf{x}_t) = e^{\mathbf{A}_i\, t}\, \mathbf{x}_t \tag{1}$$

donde $i \in \{0, 1, \ldots, N-1\}$ representa el modo, $\mathbf{A}_i$ son matrices de transformación y t es el tiempo de evolución.

2.2. Implementación Específica

En nuestra implementación utilizamos:

$$\mathbf{A}_1 = \begin{pmatrix} 1 & 50 \\ -1 & 1 \end{pmatrix}, \tag{2}$$

$$\mathbf{A}_2 = \begin{pmatrix} 1 & -1 \\ 50 & 1 \end{pmatrix}. \tag{3}$$

2.3. Función de Recompensa

La función de recompensa mide la desviación entre el estado transformado y el estado inicial:

$$r(\mathbf{x}_t, i) = -\big|\|\mathbf{x}_{t+1}\| - \|\mathbf{x}_t\|\big|, \tag{4}$$

donde $\mathbf{x}_{t+1} = e^{\mathbf{A}_i\, t}\, \mathbf{x}_t$.

3. Normalized Advantage Function (NAF)

3.1. Arquitectura de la Red

La red NAF parametriza directamente el valor Q mediante:

$$Q(\mathbf{x}, \mathbf{u}) = V(\mathbf{x}) - \tfrac{1}{2}(\mathbf{u} - \boldsymbol{\mu}(\mathbf{x}))^T \mathbf{P}(\mathbf{x})\, (\mathbf{u} - \boldsymbol{\mu}(\mathbf{x})), \tag{5}$$

donde

- $V(\mathbf{x})$: función de valor del estado,
- $\boldsymbol{\mu}(\mathbf{x})$: acción óptima,

- $\mathbf{P}(\mathbf{x}) = \mathbf{L}(\mathbf{x})\,\mathbf{L}(\mathbf{x})^T$: matriz de ventaja (definida positiva),
- $\mathbf{L}(\mathbf{x})$: matriz triangular inferior.

3.2. NAF Corregido con Exponencial de Matriz

Algorithm 1 NAF Corregido

Entrada: Estado inicial $\mathbf{x}_0$, matrices $\mathbf{A}_1, \mathbf{A}_2$, tiempo t.
Calcular $e^{\mathbf{A}_1 t}$ y $e^{\mathbf{A}_2 t}$.
for cada modo $i \in \{1, 2\}$ **do**
    $\mathbf{x}_i = e^{\mathbf{A}_i t}\,\mathbf{x}_0$.
    $r_i = -\big|\|\mathbf{x}_i\| - \|\mathbf{x}_0\|\big|$.
end for
Salida: $i^* = \arg\min_i |r_i|$.

4. Hybrid Normalized Advantage Function (HNAF)

4.1. Arquitectura Multi-Modo

El HNAF extiende el NAF para manejar múltiples modos mediante:

- **Redes separadas:** una red NAF por cada modo $Q_i(\mathbf{x}, \mathbf{u})$.
- **Redes objetivo:** Q_i^{target} para estabilización.
- **Buffers de replay:** uno por modo.

4.2. Red Neuronal Mejorada

Algorithm 2 Arquitectura de Red HNAF

Entrada: Dimensión de estado $d_s = 2$, acción $d_a = 2$.
Capas ocultas: L capas de H unidades.
for $l = 1$ **to** L **do**
    $\mathbf{h}_l = \text{ReLU}\big(\text{BatchNorm}(\mathbf{W}_l \mathbf{h}_{l-1} + \mathbf{b}_l)\big)$.
end for
$V(\mathbf{x}) = \mathbf{w}_V^T \mathbf{h}_L + b_V$.
$\boldsymbol{\mu}(\mathbf{x}) = \tanh(\mathbf{W}_\mu \mathbf{h}_L + \mathbf{b}_\mu) \times 0{,}1$.
$\mathbf{L}(\mathbf{x}) = \text{TriL}(\mathbf{W}_L \mathbf{h}_L + \mathbf{b}_L)$.

4.3. Selección de Acción

4.4. Prioritized Experience Replay

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}, \quad p_i = |\delta_i| + \varepsilon. \tag{6}$$

Algorithm 3 Selección de Acción HNAF

Entrada: Estado $\mathbf{x}$, exploración ϵ.
if random() $< \epsilon$ **then**
  Seleccionar $i \sim \text{Uniform}(\{0, \ldots, N-1\})$.
else
  for cada modo i **do**
    Calcular $Q_i(\mathbf{x}, \boldsymbol{\mu}_i(\mathbf{x}))$.
  end for
  $i^* = \arg\max_i Q_i(\mathbf{x}, \boldsymbol{\mu}_i(\mathbf{x}))$.
end if
$\mathbf{u}^* = \boldsymbol{\mu}_{i^*}(\mathbf{x})$.
Salida: $(i^*, \mathbf{u}^*)$.

Algorithm 4 Actualización HNAF con Prioritized Replay

Entrada: Batch B, parámetros α, β.
for cada modo i **do**
  if $|\text{Buffer}_i| \geq B$ **then**
    Muestrear $\{(\mathbf{x}_j, i_j, \mathbf{u}_j, r_j, \mathbf{x}'_j)\}_{j=1}^{B}$.
    for cada j **do**
      $Q_{\text{target}} = r_j + \gamma \max_{i'} Q_{i'}^{\text{target}}(\mathbf{x}'_j, \boldsymbol{\mu}_{i'}(\mathbf{x}'_j))$.
      $Q_{\text{current}} = Q_i(\mathbf{x}_j, \mathbf{u}_j)$.
      $\delta_j = Q_{\text{target}} - Q_{\text{current}}$.
    end for
    $\mathcal{L} = \frac{1}{B} \sum_j w_j \, \delta_j^2, \; w_j = (N \cdot P(j))^{-\beta}$.
    Optimizar $\mathcal{L}$.
    Actualizar $p_j = |\delta_j| + \varepsilon$.
  end if
end for
$\theta^{\text{target}} \leftarrow \tau \, \theta + (1 - \tau) \, \theta^{\text{target}}$.

5. Verificación

5.1. Estados de Prueba

Algorithm 5 Verificación por Estados

$\mathcal{S} = \{[0{,}1, 0{,}1], [0, 0{,}1], [0{,}1, 0], [0{,}05, 0{,}05], [-0{,}05, 0{,}08]\}$.
for $\mathbf{x} \in \mathcal{S}$ **do**
  $(i_{\text{HNAF}}, \mathbf{u}) = \text{HNAF.select\_action}(\mathbf{x})$.
  $i_{\text{opt}} = \text{get\_optimal\_mode}(\mathbf{x})$.
  Verificar $(i_{\text{HNAF}} = i_{\text{opt}})$.
end for
Precisión $= \frac{\#\text{correctos}}{|\mathcal{S}|}$.

5.2. Evaluación en Rejilla

Algorithm 6 Evaluación en Rejilla

$G = 50$ o 100.
Generar $\mathbf{X}, \mathbf{Y} = \text{meshgrid}(\text{linspace}(-0{,}5, 0{,}5, G))$.
correctos $= 0$, total $= G^2$.
for $i, j = 1$ **to** G **do**
  $\mathbf{x} = [X_{i,j}, Y_{i,j}]$.
  $i_{\text{HNAF}} = \text{HNAF.select\_action}(\mathbf{x}, \epsilon = 0)$.
  $i_{\text{opt}} = \text{get\_optimal\_mode}(\mathbf{x})$.
  if $i_{\text{HNAF}} = i_{\text{opt}}$ **then**
    correctos $++$.
  end if
end for
Precisión $= \frac{\text{correctos}}{\text{total}}$.

6. Métricas de Convergencia

6.1. Criterios

1. Precisión en rejilla $\geq 90\,\%$.
2. Precisión en estados de prueba $= 100\,\%$.
3. $\text{std}(\text{recompensas}_{100}) \leq 0{,}1$.
4. $\text{std}(\text{pérdidas}_{50}) \leq 0{,}01$.
5. Uso balanceado de modos $0{,}2 \leq \frac{\text{uso}_0}{\text{uso}_1} \leq 5{,}0$.

Episodio	ϵ	Recompensa	Precisión	Pérdida
50	0.475	-2.50	60 %	0.150
100	0.450	-1.80	72 %	0.095
200	0.400	-1.20	85 %	0.068
500	0.275	-0.45	92 %	0.025
750	0.163	-0.32	94 %	0.018
1000	0.050	-0.30	94.5 %	0.016

Cuadro 1: Evolución de métricas durante el entrenamiento

6.2. Progresión de Entrenamiento

7. Implementación

7.1. Parámetros de Configuración

Listing 1: Configuracion HNAF

```
# Arquitectura de red
state_dim = 2
action_dim = 2
num_modes = 2
hidden_dim = 64
num_layers = 3

# Entrenamiento
num_episodes = 1000
batch_size = 32
lr = 1e-4
gamma = 0.99
tau = 0.001

# Exploracion
initial_epsilon = 0.5
final_epsilon = 0.05

# Prioritized Replay
buffer_capacity = 10000
alpha = 0.6  # priorizacion
beta = 0.4   # correccion de sesgo

# Evaluacion
eval_interval = 50
grid_size = 50  # verificacion rapida
```

7.2. Flujo de Entrenamiento

Algorithm 7 Entrenamiento Completo HNAF

Inicializar HNAF con los parámetros.
Inicializar listas de recompensas, pérdidas y precisiones.
for episode = 1 **to** num_episodes **do**
    Actualizar ϵ.
    r = HNAF.train_episode(ϵ).
    Agregar r a recompensas.
    ℓ = HNAF.update($batch\_size$).
    if $\ell \neq$ None **then**
        Agregar ℓ a pérdidas.
    end if
    Actualizar redes objetivo.
    if episode mód $eval\_interval = 0$ **then**
        Evaluar política y rejilla; agregar precisión.
        Mostrar métricas.
    end if
end for
Verificación final: HNAF.verify_hnaf().

8. Resultados de Verificación

8.1. Ejemplo de Salida

Listing 2: Verificacion Exitosa

```
==========================================================
VERIFICACION HNAF
==========================================================
Estado [0.1, 0.1]: HNAF Modo 1, Optimo 1, Correcto
Estado [0, 0.1]:  HNAF Modo 0, Optimo 0, Correcto
Estado [0.1, 0]:  HNAF Modo 1, Optimo 1, Correcto
Estado [0.05,0.05]: HNAF Modo 1, Optimo 1, Correcto
Estado [-0.05,0.08]:HNAF Modo 0, Optimo 0, Correcto
Precision en rejilla 50x50: 94.20%
```

8.2. Indicadores de Fallo

Listing 3: Senales de Fallo

```
INDICADORES DE FALLO
- Precision en rejilla < 90%
- Estados de prueba incorrectos
- Recompensas oscilantes
- Uso de un solo modo (>95%)
- Perdidas no convergentes
```

9. Conclusiones

La implementación del HNAF ofrece:

- Arquitectura escalable a N modos.
- Verificación rigurosa en estados de prueba y rejilla.
- Convergencia certificada por múltiples métricas.
- Optimizaciones con Prioritized Replay y normalización.