neb_ap2

May 9, 2022

José Geraldo Fernandes - 2022667276

Mecanismos de Inferência Nebulosa

[1]:
```
import simpful as sf
import numpy as np
import matplotlib.pyplot as plt

from matplotlib import cm
from matplotlib.colors import ListedColormap
```

1 Aproximação de Funções

1.1 Função Quadrática

- $y = x^2,\ x \in [-2,\ 2]$

1.1.1 Mamdani

[2]:
```
FS = sf.FuzzySystem()
```

```
 _____  __  _  _  ____  ____  _  _  __
/ ___)(  )( \/ )(  _ \(  __)/ )( \(  ) v2.6.3
\___ \ )( / \/ \ ) __/ ) _) ) \/ (/ (_/\
(____/(__)\_)(_/(__)  (__)  \____/\____/

Created by Marco S. Nobile (m.s.nobile@tue.nl)
and Simone Spolaor (simone.spolaor@unimib.it)
```

Antecedentes

[3]:
```
S_1 = sf.FuzzySet(function = sf.Triangular_MF(a = -3, b = -2, c = -1), term =␣
↪'doisn')
S_2 = sf.FuzzySet(function = sf.Triangular_MF(a = -2, b = -1, c = -0.3), term =␣
↪'umn')
S_3 = sf.FuzzySet(function = sf.Triangular_MF(a = -1, b = 0, c = 1), term =␣
↪'zero')
```

```
S_4 = sf.FuzzySet(function = sf.Triangular_MF(a = 0.3, b = 1, c = 2), term =␣
↪'ump')
S_5 = sf.FuzzySet(function = sf.Triangular_MF(a = 1, b = 2, c = 3), term =␣
↪'doisp')
FS.add_linguistic_variable('x', sf.LinguisticVariable([S_1, S_2, S_3, S_4,␣
↪S_5], universe_of_discourse = [-2, 2]))
```

Consequentes

[4]:
```
T_1 = sf.FuzzySet(function = sf.Triangular_MF(a = -0.1, b = 0, c = 0.1), term =␣
↪'ZERO')
T_2 = sf.FuzzySet(function = sf.Triangular_MF(a = 0.9, b = 1, c = 1.1), term =␣
↪'UM')
T_3 = sf.FuzzySet(function = sf.Triangular_MF(a = 3.9, b = 4, c = 4.1), term =␣
↪'QUATRO')
FS.add_linguistic_variable('y', sf.LinguisticVariable([T_1, T_2, T_3],␣
↪universe_of_discourse = [0, 4]))
```

Regras

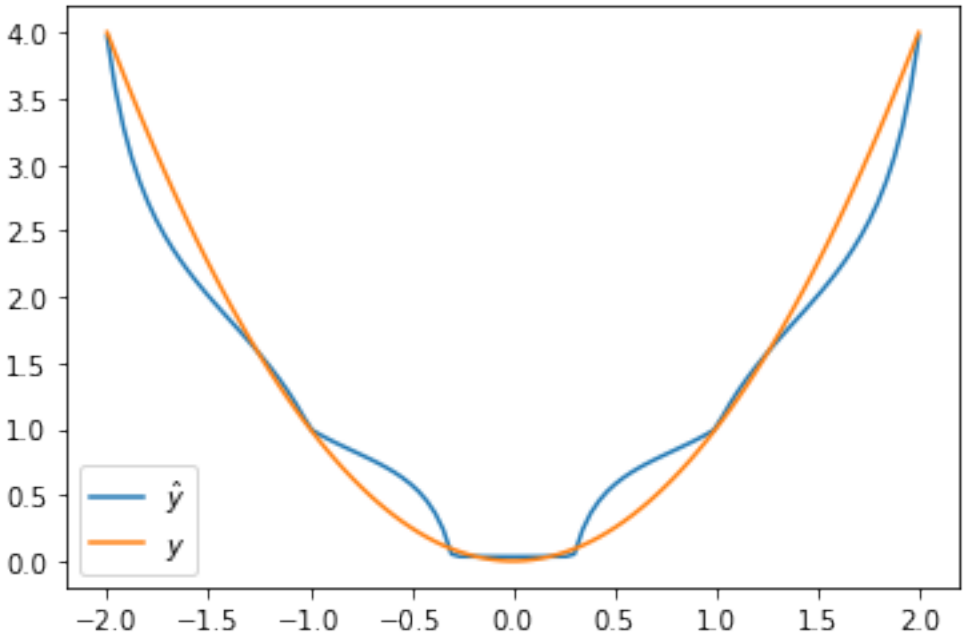
[5]:
```
RULE1 = 'IF (x IS doisn) THEN (y IS QUATRO)'
RULE2 = 'IF (x IS umn) THEN (y IS UM)'
RULE3 = 'IF (x IS zero) THEN (y IS ZERO)'
RULE4 = 'IF (x IS ump) THEN (y IS UM)'
RULE5 = 'IF (x IS doisp) THEN (y IS QUATRO)'
FS.add_rules([RULE1, RULE2, RULE3, RULE4, RULE5])
```

Síntese

[6]:
```
N = 100
X = np.linspace(start = -2, stop = 2, num = N)
y = X ** 2
yhat = []
for x in X:
    FS.set_variable('x', x)
    yhat.append(FS.inference().get('y'))
```

[7]:
```
plt.plot(X, yhat, label = r'$\hat{y}$')
plt.plot(X, y, label = r'$y$')
plt.legend()
mse = np.sqrt(np.sum((y - yhat) ** 2)) / len(yhat)
print(mse)
```

```
0.025515842576160516
```



1.1.2 Sugeno linear

[8]:
```
FS = sf.FuzzySystem()
```

```
 _____  __  _  _  ____  ____  _  _  __
/ ___)(  )( \/ )(  _ \(  __)/ )( \(  ) v2.6.3
\___ \ )( / \/ \ ) __/ ) _) ) \/ (/ (_/\
(____/(__)\_)(_/(__)  (__)  \____/\____/

Created by Marco S. Nobile (m.s.nobile@tue.nl)
and Simone Spolaor (simone.spolaor@unimib.it)
```

Antecedentes

[9]:
```
S_1 = sf.FuzzySet(function = sf.Gaussian_MF(mu = -2, sigma = 2), term =␣
↪'negativo')
S_2 = sf.FuzzySet(function = sf.Gaussian_MF(mu = 2, sigma = 2), term =␣
↪'positivo')
FS.add_linguistic_variable('x', sf.LinguisticVariable([S_1, S_2],␣
↪universe_of_discourse = [-2, 2]))
```

Consequentes

[10]:
```
FS.set_output_function('decrescente', '-2*x')
FS.set_output_function('crescente', '2*x')
```

```
 * Detected Sugeno model type
```

Regras

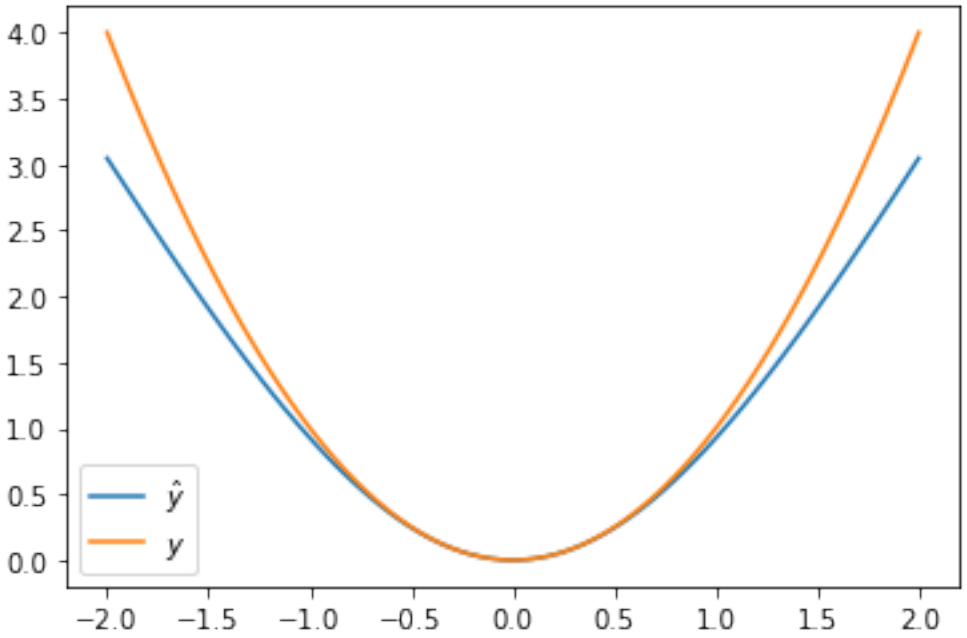
[11]:
```
RULE1 = 'IF (x IS negativo) THEN (y IS decrescente)'
RULE2 = 'IF (x IS positivo) THEN (y IS crescente)'
FS.add_rules([RULE1, RULE2])
```

Síntese

[12]:
```
N = 100
X = np.linspace(start = -2, stop = 2, num = N)
y = X ** 2
yhat = []
for x in X:
    FS.set_variable('x', x)
    yhat.append(FS.Sugeno_inference(['y']).get('y'))
```

[13]:
```
plt.plot(X, yhat, label = r'$\hat{y}$')
plt.plot(X, y, label = r'$y$')
plt.legend()
mse = np.sqrt(np.sum((y - yhat) ** 2)) / len(yhat)
print(mse)
```

```
0.03482340454322335
```



1.1.3 Sugeno constante

[14]:
```
FS = sf.FuzzySystem()
```

```
 ____  __  _  _  ____  ____  _  _  __
/ ___)(  )( \/ )(  _ \(  __)/ )( \(  ) v2.6.3
\___ \ )( / \/ \ ) __/ ) _) ) \/ (/ (_/\
(____/(__)\_)(_/(__)  (__)  \____/\____/

Created by Marco S. Nobile (m.s.nobile@tue.nl)
and Simone Spolaor (simone.spolaor@unimib.it)
```

Antecedentes

[15]:
```
S_1 = sf.FuzzySet(function = sf.Triangular_MF(a = -3, b = -2, c = -1), term =␣
↪'doisn')
S_2 = sf.FuzzySet(function = sf.Triangular_MF(a = -2, b = -1, c = 0), term =␣
↪'umn')
S_3 = sf.FuzzySet(function = sf.Triangular_MF(a = -1, b = 0, c = 1), term =␣
↪'zero')
S_4 = sf.FuzzySet(function = sf.Triangular_MF(a = 0, b = 1, c = 2), term =␣
↪'ump')
S_5 = sf.FuzzySet(function = sf.Triangular_MF(a = 1, b = 2, c = 3), term =␣
↪'doisp')
FS.add_linguistic_variable('x', sf.LinguisticVariable([S_1, S_2, S_3, S_4,␣
↪S_5], universe_of_discourse = [-2, 2]))
```

Consequentes

[16]:
```
FS.set_crisp_output_value('ZERO', 0)
FS.set_crisp_output_value('UM', 1)
FS.set_crisp_output_value('QUATRO', 4)
```

```
 * Detected Sugeno model type
```

Regras

[17]:
```
RULE1 = 'IF (x IS doisn) THEN (y IS QUATRO)'
RULE2 = 'IF (x IS umn) THEN (y IS UM)'
RULE3 = 'IF (x IS zero) THEN (y IS ZERO)'
RULE4 = 'IF (x IS ump) THEN (y IS UM)'
RULE5 = 'IF (x IS doisp) THEN (y IS QUATRO)'
FS.add_rules([RULE1, RULE2, RULE3, RULE4, RULE5])
```

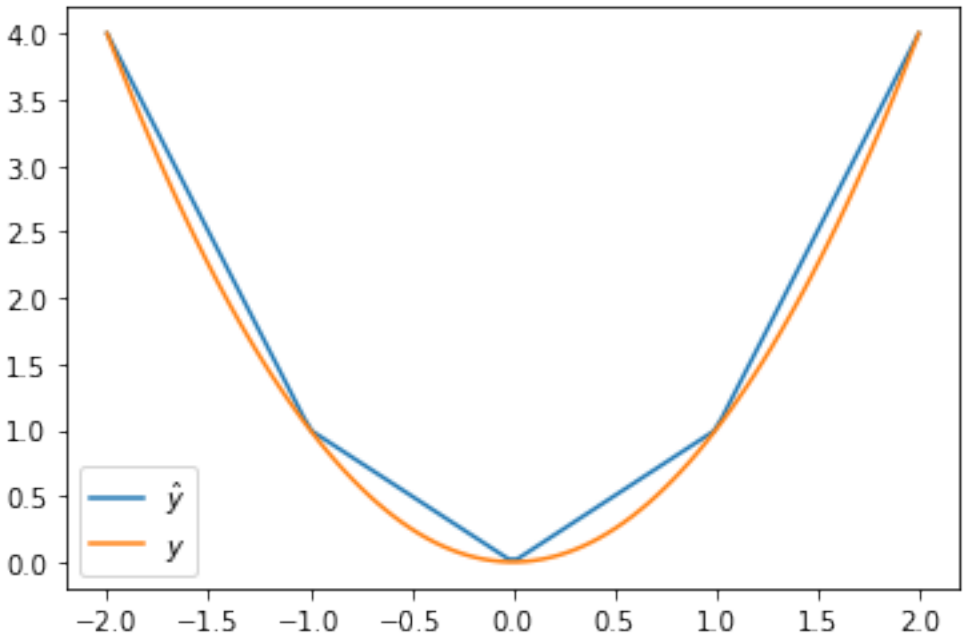
Síntese

[18]:
```
N = 100
X = np.linspace(start = -2, stop = 2, num = N)
y = X ** 2
```

```
yhat = []
for x in X:
    FS.set_variable('x', x)
    yhat.append(FS.Sugeno_inference(['y']).get('y'))
```

[19]:
```
plt.plot(X, yhat, label = r'$\hat{y}$')
plt.plot(X, y, label = r'$y$')
plt.legend()
mse = np.sqrt(np.sum((y - yhat) ** 2)) / len(yhat)
print(mse)
```

```
0.018165902030029 58
```



1.2 Função Sinc

- $y = \text{sinc}(x),\ x \in [0,\ 2\pi]$

1.2.1 Mamdani

[20]:
```
FS = sf.FuzzySystem()
```

```
 ____  __  _  _  ____  ____  _  _  __
/ ___)(  )( \/ )(  _ \(  __)/ )( \(  ) v2.6.3
\___ \ )( / \/ \ ) __/ ) _) ) \/ (/ (_/\
(____/(__)\_)(_/(__)  (__)  \____/\____/
```

```
Created by Marco S. Nobile (m.s.nobile@tue.nl)
and Simone Spolaor (simone.spolaor@unimib.it)
```

Antecedentes

[21]:
```
S_1 = sf.FuzzySet(function = sf.Triangular_MF(a = -1, b = 0, c = 4), term =␣
↪'maximo')
S_2 = sf.FuzzySet(function = sf.Triangular_MF(a = 0, b = 4.5, c = 8), term =␣
↪'minimo')
FS.add_linguistic_variable('x', sf.LinguisticVariable([S_1, S_2],␣
↪universe_of_discourse = [0, 2*np.pi]))
```

Consequentes

[22]:
```
T_1 = sf.FuzzySet(function = sf.Triangular_MF(a = -0.62, b = -0.22, c = -0.12),␣
↪term = 'MINIMO')
T_2 = sf.FuzzySet(function = sf.Triangular_MF(a = 0.9, b = 1, c = 1.1), term =␣
↪'MAXIMO')
FS.add_linguistic_variable('y', sf.LinguisticVariable([T_1, T_2],␣
↪universe_of_discourse = [-0.22, 1]))
```

Regras

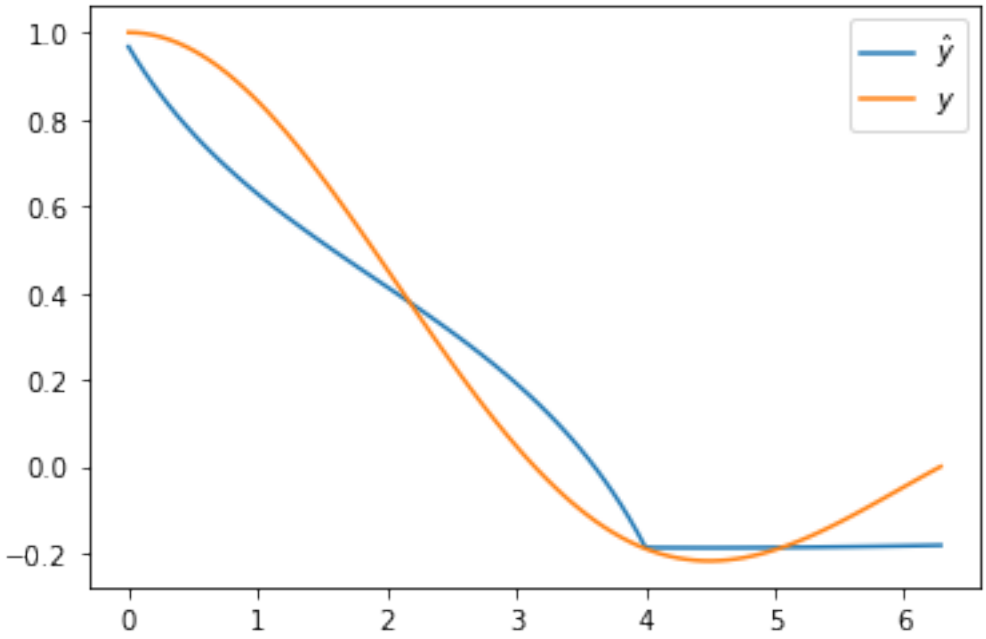
[23]:
```
RULE1 = 'IF (x IS maximo) THEN (y IS MAXIMO)'
RULE2 = 'IF (x IS minimo) THEN (y IS MINIMO)'
FS.add_rules([RULE1, RULE2])
```

Síntese

[24]:
```
N = 100
X = np.linspace(start = 1e-4, stop = 2 * np.pi, num = N)
y = np.sin(X) / X
yhat = []
for x in X:
    FS.set_variable('x', x)
    yhat.append(FS.inference().get('y'))
```

[25]:
```
plt.plot(X, yhat, label = r'$\hat{y}$')
plt.plot(X, y, label = r'$y$')
plt.legend()
mse = np.sqrt(np.sum((y - yhat) ** 2)) / len(yhat)
print(mse)
```

```
0.011994798107316655
```



1.2.2 Sugeno linear

[26]:
```
FS = sf.FuzzySystem()
```

```
 _____  __  _  _  ____  ____  _  _  __
/ ___)(  )( \/ )(  _ \(  __)/ )( \(  ) v2.6.3
\___ \ )( / \/ \ ) __/ ) _) ) \/ (/ (_/\
(____/(__)\_)(_/(__)  (__)  \____/\____/

Created by Marco S. Nobile (m.s.nobile@tue.nl)
and Simone Spolaor (simone.spolaor@unimib.it)
```

Antecedentes

[27]:
```
S_1 = sf.FuzzySet(function = sf.Triangular_MF(a = 0, b = 1.5, c = 5.2), term =␣
↪'queda')
S_2 = sf.FuzzySet(function = sf.Triangular_MF(a = 1.5, b = 5.2, c = 2*np.pi),␣
↪term = 'subida')
FS.add_linguistic_variable('x', sf.LinguisticVariable([S_1, S_2],␣
↪universe_of_discourse = [0, 2*np.pi]))
```

Consequentes

[28]:
```
FS.set_output_function('decrescente', '-0.4*x+1.26')
FS.set_output_function('crescente', '0.12*x-0.79')
```

```
 * Detected Sugeno model type
```

Regras

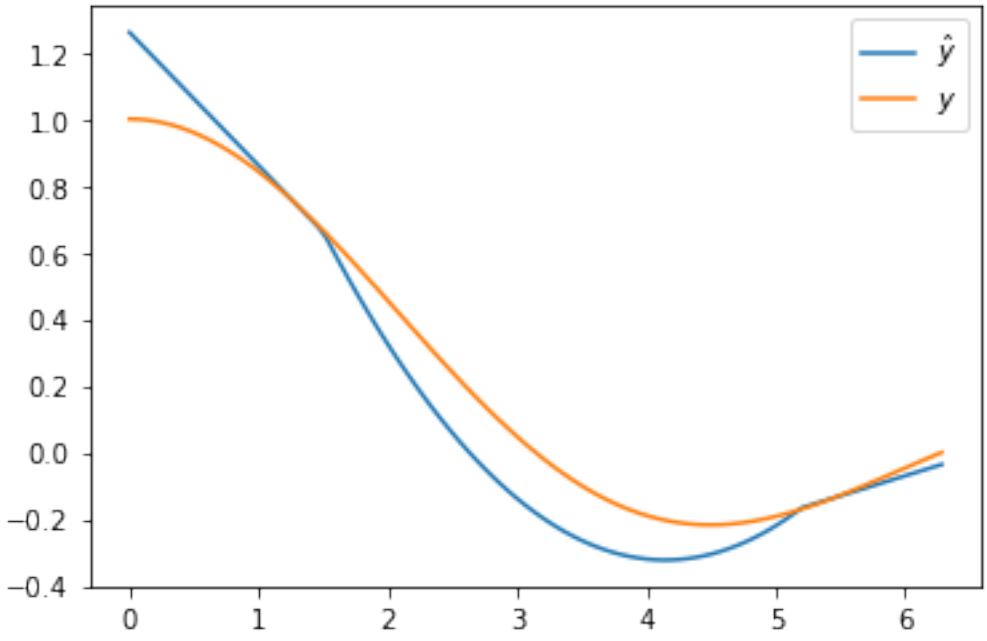
[29]:
```
RULE1 = 'IF (x IS queda) THEN (y IS decrescente)'
RULE2 = 'IF (x IS subida) THEN (y IS crescente)'
FS.add_rules([RULE1, RULE2])
```

Síntese

[30]:
```
N = 100
X = np.linspace(start = 1e-4, stop = 2*np.pi-1e-4, num = N)
y = np.sin(X) / X
yhat = []
for x in X:
    FS.set_variable('x', x)
    yhat.append(FS.Sugeno_inference(['y']).get('y'))
```

[31]:
```
plt.plot(X, yhat, label = r'$\hat{y}$')
plt.plot(X, y, label = r'$y$')
plt.legend()
mse = np.sqrt(np.sum((y - yhat) ** 2)) / len(yhat)
print(mse)
```

```
0.011865047511214984
```



1.2.3 Sugeno constante

[32]:
```
FS = sf.FuzzySystem()
```

```
 ____  __  _  _  ____  ____  _  _  __
/ ___)(  )( \/ )(  _ \(  __)/ )( \(  ) v2.6.3
\___ \ )( / \/ \ ) __/ ) _) ) \/ (/ (_/\
(____/(__)\_)(_/(__)  (__)  \____/\____/

Created by Marco S. Nobile (m.s.nobile@tue.nl)
and Simone Spolaor (simone.spolaor@unimib.it)
```

Antecedentes

[33]:
```
S_1 = sf.FuzzySet(function = sf.Triangular_MF(a = -1, b = 0, c = np.pi), term =␣
↪'maximo')
S_2 = sf.FuzzySet(function = sf.Triangular_MF(a = 0, b = np.pi, c = 4.5), term␣
↪= 'rooti')
S_3 = sf.FuzzySet(function = sf.Triangular_MF(a = np.pi, b = 4.5, c = 2*np.pi),␣
↪term = 'minimo')
S_4 = sf.FuzzySet(function = sf.Triangular_MF(a = 4.5, b = 2*np.pi, c = 2*np.
↪pi+1), term = 'roots')
FS.add_linguistic_variable('x', sf.LinguisticVariable([S_1, S_2, S_3, S_4],␣
↪universe_of_discourse = [0, 2*np.pi]))
```

Consequentes

[34]:
```
FS.set_crisp_output_value('MAXIMO', 1)
FS.set_crisp_output_value('ZERO', 0)
FS.set_crisp_output_value('MINIMO', -0.22)
```

```
 * Detected Sugeno model type
```

Regras

[35]:
```
RULE1 = 'IF (x IS maximo) THEN (y IS MAXIMO)'
RULE2 = 'IF (x IS rooti) THEN (y IS ZERO)'
RULE3 = 'IF (x IS minimo) THEN (y IS MINIMO)'
RULE4 = 'IF (x IS roots) THEN (y IS ZERO)'
FS.add_rules([RULE1, RULE2, RULE3, RULE4])
```

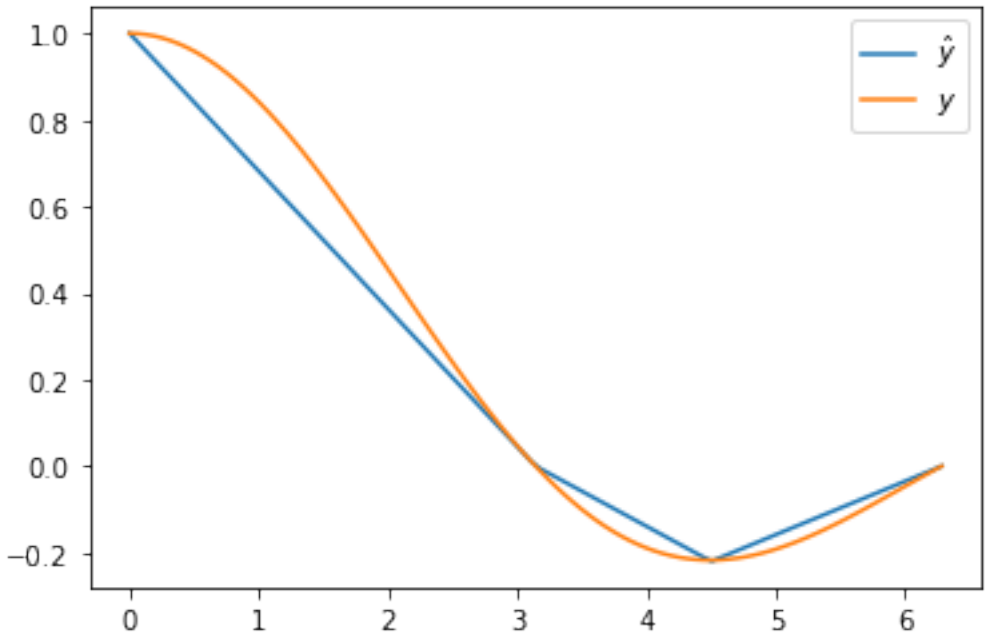
Síntese

[36]:
```
N = 100
X = np.linspace(start = 1e-4, stop = 2 * np.pi, num = N)
y = np.sin(X) / X
yhat = []
for x in X:
    FS.set_variable('x', x)
```

```
    yhat.append(FS.Sugeno_inference().get('y'))
```

[37]:
```
plt.plot(X, yhat, label = r'$\hat{y}$')
plt.plot(X, y, label = r'$y$')
plt.legend()
mse = np.sqrt(np.sum((y - yhat) ** 2)) / len(yhat)
print(mse)
```

```
0.007732502485169644
```



2 Classificação de Padrões

[38]:
```
FS = sf.FuzzySystem()
```

```
 ____  __  _  _  ____  ____  _  _  __
/ ___)(  )( \/ )(  _ \(  __)/ )( \(  ) v2.6.3
\___ \ )( / \/ \ ) __/ ) _) ) \/ (/ (_/\
(____/(__)\_)(_/(__)  (__)  \____/\____/

Created by Marco S. Nobile (m.s.nobile@tue.nl)
and Simone Spolaor (simone.spolaor@unimib.it)
```

Antecedentes

[39]:
```
S_1 = sf.FuzzySet(function = sf.Gaussian_MF(mu = 0, sigma = 0.2), term =␣
↪'xazul')
S_2 = sf.FuzzySet(function = sf.Gaussian_MF(mu = -1, sigma = 0.2), term =␣
↪'xpreto')
S_3 = sf.FuzzySet(function = sf.Gaussian_MF(mu = 1, sigma = 0.2), term =␣
↪'xlaranja')
S_4 = sf.FuzzySet(function = sf.Gaussian_MF(mu = 1, sigma = 0.2), term =␣
↪'xroxo')
S_5 = sf.FuzzySet(function = sf.Gaussian_MF(mu = -1, sigma = 0.2), term =␣
↪'xamarelo')
FS.add_linguistic_variable('x', sf.LinguisticVariable([S_1, S_2, S_3, S_4,␣
↪S_5], universe_of_discourse = [-2, 2]))
```

[40]:
```
T_1 = sf.FuzzySet(function = sf.Gaussian_MF(mu = 0, sigma = 0.2), term =␣
↪'yazul')
T_2 = sf.FuzzySet(function = sf.Gaussian_MF(mu = 1, sigma = 0.2), term =␣
↪'ypreto')
T_3 = sf.FuzzySet(function = sf.Gaussian_MF(mu = 1, sigma = 0.2), term =␣
↪'ylaranja')
T_4 = sf.FuzzySet(function = sf.Gaussian_MF(mu = -1, sigma = 0.2), term =␣
↪'yroxo')
T_5 = sf.FuzzySet(function = sf.Gaussian_MF(mu = -1, sigma = 0.2), term =␣
↪'yamarelo')
FS.add_linguistic_variable('y', sf.LinguisticVariable([T_1, T_2, T_3, T_4,␣
↪T_5], universe_of_discourse = [-2, 2]))
```

Consequentes

[41]:
```
FS.set_crisp_output_value('AZUL', 1)
FS.set_crisp_output_value('PRETO', 2)
FS.set_crisp_output_value('LARANJA', 3)
FS.set_crisp_output_value('ROXO', 4)
FS.set_crisp_output_value('AMARELO', 5)
```

```
 * Detected Sugeno model type
```

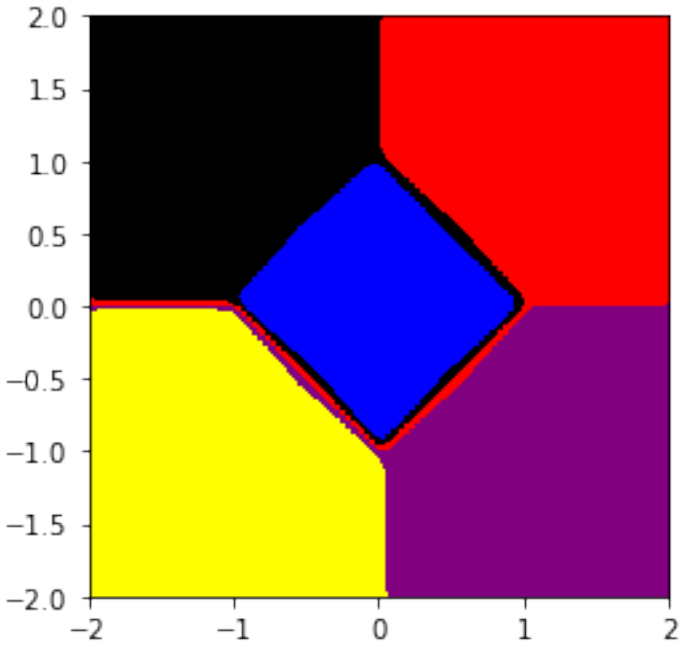
Regras

[42]:
```
RULE1 = 'IF (x IS xazul) AND (y IS yazul) THEN (z IS AZUL)'
RULE2 = 'IF (x IS xpreto) AND (y IS ypreto) THEN (z IS PRETO)'
RULE3 = 'IF (x IS xlaranja) AND (y IS ylaranja) THEN (z IS LARANJA)'
RULE4 = 'IF (x IS xroxo) AND (y IS yroxo) THEN (z IS ROXO)'
RULE5 = 'IF (x IS xamarelo) AND (y IS yamarelo) THEN (z IS AMARELO)'
FS.add_rules([RULE1, RULE2, RULE3, RULE4, RULE5])
```

Síntese

[43]:
```
N = 100
xx = np.linspace(-2, 2, N)
yy = np.linspace(-2, 2, N)
yy[::-1].sort()
XX, YY = np.meshgrid(xx, yy)
Z = np.zeros(shape = XX.shape)
for i in range(N):
    for j in range(N):
        x = XX[i, j]
        y = YY[i, j]
        FS.set_variable('x', x)
        FS.set_variable('y', y)
        Z[i, j] = FS.Sugeno_inference(['z']).get('z')
```

[44]:
```
cmap = ListedColormap(["blue", "black", "red", "purple", "yellow"])
plt.imshow(Z, cmap = cmap, extent = [-2, 2, -2, 2]);
```



Comparando com a superfície teórica:

$f_i = [(2\pi)^k|\ |]^{-\frac{1}{2}} \exp(-\frac{1}{2}(\mathbf{x} - \mu_i)^T\ ^{-1}(\mathbf{x} - \mu_i))$

Seja a superfície r_{ij}, $\mathbf{x}_{ij} \in r_{ij}$ a intersecção entre as classes i e j:

$f_i(\mathbf{x}_{ij}) = f_j(\mathbf{x}_{ij})$

$(\mathbf{x}_{ij} - \mu_i)^T\ ^{-1}(\mathbf{x}_{ij} - \mu_i) = (\mathbf{x}_{ij} - \mu_j)^T\ ^{-1}(\mathbf{x}_{ij} - \mu_j)$

$(\mathbf{x}_{ij} - \mu_i)^T\sigma(\mathbf{x}_{ij} - \mu_i) = (\mathbf{x}_{ij} - \mu_j)^T\sigma(\mathbf{x}_{ij} - \mu_j)$

$(\mathbf{x}_{ij} - \mu_i)^T(\mathbf{x}_{ij} - \mu_i) = (\mathbf{x}_{ij} - \mu_j)^T(\mathbf{x}_{ij} - \mu_j)$

$(x_{ij} - \mu_{ix})^2 + (y_{ij} - \mu_{iy})^2 = (x_{ij} - \mu_{jx})^2 + (y_{ij} - \mu_{jy})^2$

$y_{ij}(-\mu_{iy} + \mu_{jy}) = x_{ij}(\mu_{ix} - \mu_{jx}) + \sum_i \sum_j \mu_{ij}^2(-1)^i$

$y_{ij}K = (Ax_{ij} + B)$

Computando para todos os pares:

[45]:
```
def yij(ui, uj):
    K = -ui[1] + uj[1]
    A = ui[0] - uj[0]
    B = -ui[0]**2 - ui[1]**2 + uj[0]**2 + uj[1]**2
    return K, A, B
```

[46]:
```
u1 = [0, 0]
u2 = [-1, 1]
u3 = [1, 1]
u4 = [1, -1]
u5 = [-1, -1]

k12, a12, b12 = yij(u1, u2)
k13, a13, b13 = yij(u1, u3)
k14, a14, b14 = yij(u1, u4)
k15, a15, b15 = yij(u1, u5)

k23, a23, b23 = yij(u2, u3)
k34, a34, b34 = yij(u3, u4)
k45, a45, b45 = yij(u4, u5)
k52, a52, b52 = yij(u5, u2)

fronteiras = [[k12, a12, b12],
[k13, a13, b13],
[k14, a14, b14],
[k15, a15, b15],
[k23, a23, b23],
[k34, a34, b34],
[k45, a45, b45],
[k52, a52, b52]]
```

[47]:
```
tol = 0.05
N = 100
xx = np.linspace(-2, 2, N)
yy = np.linspace(-2, 2, N)
yy[::-1].sort()
XX, YY = np.meshgrid(xx, yy)
Z = np.ones(shape = XX.shape)
for i in range(N):
```

```
    for j in range(N):
        x = XX[i, j]
        y = YY[i, j]
        for k, a, b in fronteiras:
            if np.abs(k*y - (a*x + b)) < tol:
                Z[i, j] = 0
```

[48]:
```
plt.imshow(Z, cmap = 'gray', extent = [-2, 2, -2, 2]);
```