In [ ]:
```
# FFR135 HW3
## Self organising map
```

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import scienceplots
plt.style.use(['science','ieee'])
```

In [ ]:
```
# load data
inputData = np.loadtxt('/Users/felixwaldschock/Library/CloudStorage/OneDrive-Chalmers/02_Courses/02_
labesData = np.loadtxt('/Users/felixwaldschock/Library/CloudStorage/OneDrive-Chalmers/02_Courses/02_
```

In [ ]:
```
# normalize data
inputData /= np.max(inputData)
# define parameters
input_size = inputData.shape[1]
output_size = 40
epochs = 30
batch_size = len(inputData)
eta_0 = 0.1
d_eta = 0.01
sigma_0 = 10
d_sigma = 0.05
labels = ['Class 1', 'Class 2', 'Class 3']
```

In [ ]:
```
# init the weights with random gaussian
weights = np.random.uniform(0, 1, (output_size, output_size, input_size))

def neighbourhood_function(r_i, r_i0, sigma): # openTA
    return np.exp(-0.5 * (np.linalg.norm(r_i - r_i0)**2) / sigma**2)

def get_winning_neuron(data, weights):
    distances = np.linalg.norm(weights - data, axis=2)
    # print(distances.shape)
    # print(np.argmin(distances))
    # print(np.unravel_index(np.argmin(distances), distances.shape))
    return np.unravel_index(np.argmin(distances), distances.shape)
```

In [ ]:
```

bestInitNeuron = np.zeros((len(inputData), 2))
for i in range(len(inputData)):
    bestInitNeuron[i] = get_winning_neuron(inputData[i], weights)


# train the network
for e in range(epochs):
    eta = eta_0 * np.exp(-e * d_eta)
    sigma = sigma_0 * np.exp(-e * d_sigma)

    for i in range(len(inputData)):
        # get the winning neuron
        winning_neuron = get_winning_neuron(inputData[i], weights)
        # update the weights
        for j in range(output_size):
            for k in range(output_size):
                h = neighbourhood_function(np.array([j, k]), np.array(winning_neuron), sigma)
                weights[j, k, :] += eta * h * (inputData[i] - weights[j, k, :])

bestFinalNeuron = np.zeros((len(inputData), 2))
for i in range(len(inputData)):
    bestFinalNeuron[i] = get_winning_neuron(inputData[i], weights)
```

In [ ]:

```
fig, ax = plt.subplots(1, 2, figsize=(10, 5))
# plot the initial state
ax[0].scatter(bestInitNeuron[:, 0], bestInitNeuron[:, 1], c=labesData, cmap='viridis', label=labels
ax[0].set_title('Initital state')
ax[0].set_xlabel('x')
ax[0].set_ylabel('y')



# plot
ax[1].scatter(bestFinalNeuron[:, 0], bestFinalNeuron[:, 1], c=labesData, cmap='viridis', label=labe
ax[1].set_title('Final state')
ax[1].set_xlabel('x')
ax[1].set_ylabel('y')

plt.show()
```