NIThePP_mini_school_L3-variational_classifier

September 23, 2020

1 Variational quantum classifier with Qiskit

In this notebook, we build the variational quantum classifier using Qiskit

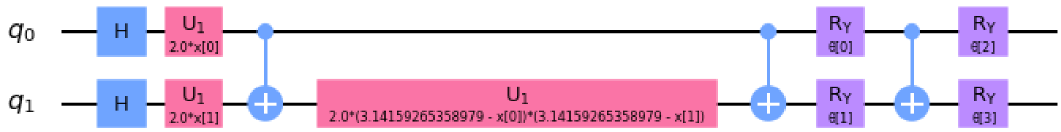
In [44]:
```
from qiskit.ml.datasets import *
from qiskit import QuantumCircuit
from qiskit.aqua.components.optimizers import COBYLA
from qiskit.circuit.library import ZZFeatureMap, RealAmplitudes
import numpy as np
import matplotlib.pyplot as plt
from qiskit.quantum_info import Statevector
%matplotlib inline
```

In [45]:
```
# size of training data set
training_size = 100

# size of test data set
test_size = 20

# dimension of data sets
n = 2

# construct training and test data
_, training_input, test_input, class_labels = \
      ad_hoc_data(training_size=training_size, test_size=test_size, n=n, gap=0.3, pl

print(class_labels)
```

```
['A', 'B']
```

In [47]:
```
sv = Statevector.from_label('0' * n)
feature_map = ZZFeatureMap(n, reps=1)
var_form = RealAmplitudes(n, reps=1)
circuit = feature_map.combine(var_form)
circuit.draw(output='mpl')
```

Out[47]:



In [48]:
```python
def get_data_dict(params, x):
    parameters = {}
    for i, p in enumerate(feature_map.ordered_parameters):
        parameters[p] = x[i]
    for i, p in enumerate(var_form.ordered_parameters):
        parameters[p] = params[i]
    return parameters
```

In [49]:
```python
data = [0.1, 1.2]
params = np.array([0.1, 1.2, 0.02, 0.1])
circ_ = circuit.assign_parameters(get_data_dict(params, data))
circ_.draw(plot_barriers=True)
```

Out[49]:
```
q_0:  H  U1(0.2)  RY(0.1)  RY(0.02)

q_1:  H  U1(2.4)  X  U1(11.811)  X  RY(1.2)  X  RY(0.1)
```

In [50]:
```python
def assign_label(bit_string, class_labels):
    hamming_weight = sum([int(k) for k in list(bit_string)])
    is_odd_parity = hamming_weight & 1
    if is_odd_parity:
        return class_labels[1]
    else:
        return class_labels[0]
```

In [51]:
```python
def return_probabilities(counts, class_labels):
    shots = sum(counts.values())
    result = {class_labels[0]: 0,
              class_labels[1]: 0}
    for key, item in counts.items():
        label = assign_label(key, class_labels)
        result[label] += counts[key]/shots
    return result
```

In [53]:
```python
return_probabilities({'00' : 10, '01': 10, '11': 20}, class_labels)
```

Out[53]:
```
{'A': 0.75, 'B': 0.25}
```

In [54]:
```
def classify(x_list, params, class_labels):
    qc_list = []
    for x in x_list:
        circ_ = circuit.assign_parameters(get_data_dict(params, x))
        qc = sv.evolve(circ_)
        qc_list += [qc]
    probs = []
    for qc in qc_list:
        counts = qc.to_counts()
        prob = return_probabilities(counts, class_labels)
        probs += [prob]
    return probs
```

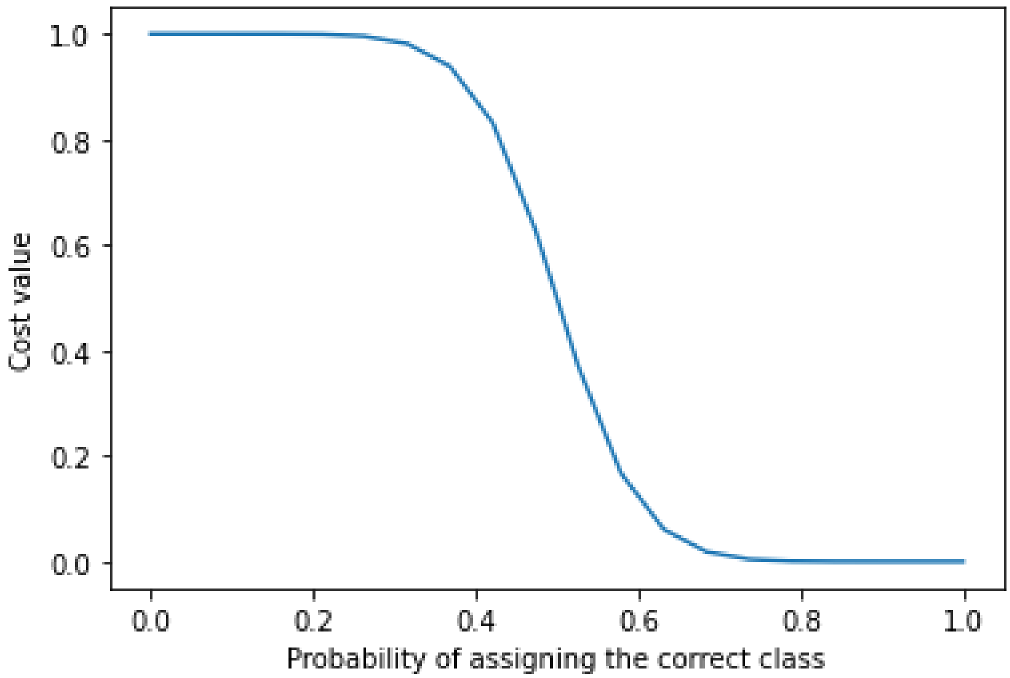
In [56]:
```
# classify a test data point
x = np.asarray([[0.5, 0.9]])
classify(x, params=np.array([0.8, -0.5, 1.5, 0,5]), class_labels=class_labels)
```

```
/Users/amyami187/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:9: DeprecationWarr
  if __name__ == '__main__':
```

Out[56]: [{'A': 0.9058187060399014, 'B': 0.09418129396009856}]

In [57]:
```
def cost_estimate_sigmoid(probs, expected_label): # probability of labels vs actual la
    p = probs.get(expected_label)
    sig = None
    if np.isclose(p, 0.0):
        sig = 1
    elif np.isclose(p, 1.0):
        sig = 0
    else:
        denominator = np.sqrt(2*p*(1-p))
        x = np.sqrt(200)*(0.5-p)/denominator
        sig = 1/(1+np.exp(-x))
    return sig
```

In [58]:
```
x = np.linspace(0, 1, 20)
y = [cost_estimate_sigmoid({'A': x_, 'B': 1-x_}, 'A') for x_ in x]
plt.plot(x, y)
plt.xlabel('Probability of assigning the correct class')
plt.ylabel('Cost value')
plt.show()
```




```
In [59]: def cost_function(training_input, class_labels, params, shots=100, print_value=False)

             # map training input to list of labels and list of samples
             cost = 0
             training_labels = []
             training_samples = []
             for label, samples in training_input.items():
                 for sample in samples:
                     training_labels += [label]
                     training_samples += [sample]

             # classify all samples
             probs = classify(training_samples, params, class_labels)

             # evaluate costs for all classified samples
             for i, prob in enumerate(probs):
                 cost += cost_estimate_sigmoid(prob, training_labels[i])
             cost /= len(training_samples)

             # print resulting objective function
             if print_value:
                 print('%.4f' % cost)

             # return objective value
             return cost
```

In [60]: cost_function(training_input, class_labels, params)

```
/Users/amyami187/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:9: DeprecationWarr
  if __name__ == '__main__':
```

Out[60]: 0.4724044012224532

1.1 Train the classifier

Training the classifier corresponds to an optimisation task. We want to minimize the cost value (sigmoid function) such that the classifier manages to properly label the given data.

In [42]:
```python
# setup the optimizer
optimizer = COBYLA(maxiter=100)

# define objective function for training
objective_function = lambda params: cost_function(training_input, class_labels, params

# randomly initialize the parameters
np.random.seed(137)
init_params = 2*np.pi*np.random.rand(n*(1)*2)

# train classifier
opt_params, value, _ = optimizer.optimize(len(init_params), objective_function, initia

# print results
print()
print('opt_params:', opt_params)
print('opt_value: ', value)
```

```
/Users/amyami187/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:9: DeprecationWarr
  if __name__ == '__main__':


0.6482
0.6274
0.6641
0.6720
0.5261
0.4336
0.4122
0.5549
0.3748
0.3038
0.2953
0.2580
0.2875
0.2569
```

0.2690
0.2814
0.2599
0.2561
0.2560
0.2505
0.2443
0.2466
0.2476
0.2391
0.2376
0.2383
0.2363
0.2358
0.2361
0.2369
0.2335
0.2332
0.2325
0.2312
0.2321
0.2312
0.2313
0.2314
0.2312
0.2313
0.2314
0.2313
0.2311
0.2311
0.2311
0.2311
0.2312
0.2310
0.2310
0.2310
0.2311
0.2311
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310

```
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310
0.2310

opt_params: [ 6.17854949 -1.56618546  3.7850562   7.21506729]
opt_value:  0.2310243542945879
```

1.2 Test the trained classifier

To check how well we could train the classifier, we evaluate the classification performance on the test data set.

In [61]:
```
# collect coordinates of test data
test_label_0_x = [x[0] for x in test_input[class_labels[0]]]
test_label_0_y = [x[1] for x in test_input[class_labels[0]]]
test_label_1_x = [x[0] for x in test_input[class_labels[1]]]
test_label_1_y = [x[1] for x in test_input[class_labels[1]]]

# initialize lists for misclassified datapoints
test_label_misclassified_x = []
test_label_misclassified_y = []
```

```python
        # evaluate test data
        for label, samples in test_input.items():

            # classify samples
            results = classify(samples, opt_params, class_labels)

            # analyze results
            for i, result in enumerate(results):

                # assign label
                assigned_label = class_labels[np.argmax([p for p in result.values()])]
                print('-----------------------------------------------------')
                print('Data point:    ', samples[i])
                print('Label:         ', label)
                print('Assigned:      ', assigned_label)
                print('Probabilities: ', result)

                if label != assigned_label:
                    print('Classification:', 'INCORRECT')
                    test_label_misclassified_x += [samples[i][0]]
                    test_label_misclassified_y += [samples[i][1]]
                else:
                    print('Classification:', 'CORRECT')

        # compute fraction of misclassified samples
        total = len(test_label_0_x) + len(test_label_1_x)
        num_misclassified = len(test_label_misclassified_x)
        print()
        print(100*(1-num_misclassified/total), "% of the test data was correctly classified!"

        # plot results
        plt.figure()
        plt.scatter(test_label_0_x, test_label_0_y, c='b', label=class_labels[0], linewidths=
        plt.scatter(test_label_1_x, test_label_1_y, c='g', label=class_labels[1], linewidths=
        plt.scatter(test_label_misclassified_x, test_label_misclassified_y, linewidths=20, s=
                    edgecolors='r')
        plt.legend()
        plt.show()
```

```
/Users/amyami187/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:9: DeprecationWar
  if __name__ == '__main__':


----------------------------------------------------
Data point:     [1.50796447 2.38761042]
Label:          A
Assigned:       B
```

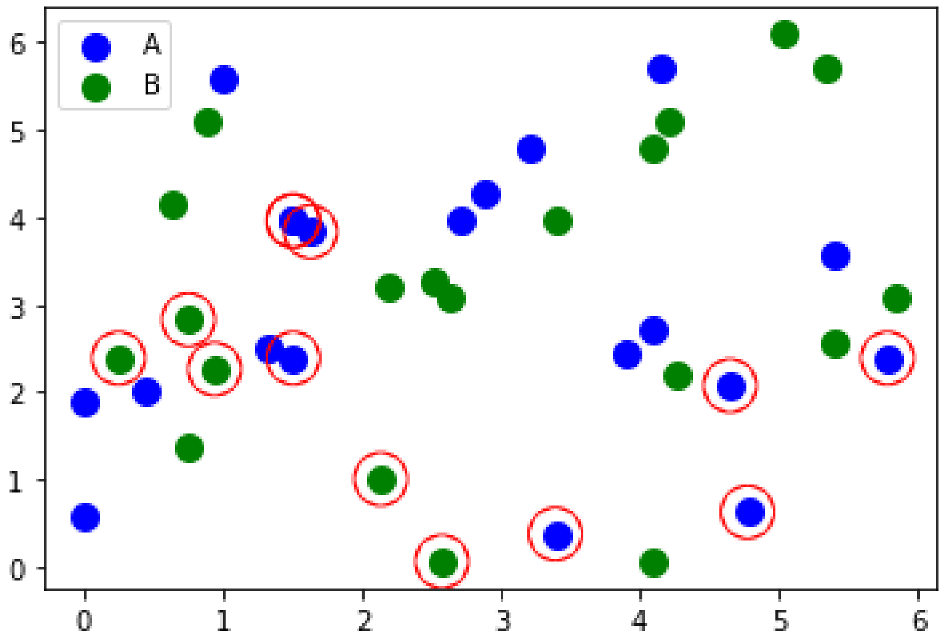
```
Probabilities:  {'A': 0.35255620717433067, 'B': 0.6474437928256694}
Classification: INCORRECT
----------------------------------------------------
Data point:     [5.78053048 2.38761042]
Label:          A
Assigned:       B
Probabilities:  {'A': 0.42519204105111874, 'B': 0.5748079589488813}
Classification: INCORRECT
----------------------------------------------------
Data point:     [3.89557489 2.45044227]
Label:          A
Assigned:       A
Probabilities:  {'A': 0.6150982038780646, 'B': 0.3849017961219354}
Classification: CORRECT
----------------------------------------------------
Data point:     [3.20442451 4.77522083]
Label:          A
Assigned:       A
Probabilities:  {'A': 0.5026198367554752, 'B': 0.49738016324452494}
Classification: CORRECT
----------------------------------------------------
Data point:     [3.39292007 0.37699112]
Label:          A
Assigned:       B
Probabilities:  {'A': 0.3570637381379921, 'B': 0.642936261862008}
Classification: INCORRECT
----------------------------------------------------
Data point:     [4.1469023  5.71769863]
Label:          A
Assigned:       A
Probabilities:  {'A': 0.6183305669871201, 'B': 0.38166943301288}
Classification: CORRECT
----------------------------------------------------
Data point:     [1.50796447 3.95840674]
Label:          A
Assigned:       B
Probabilities:  {'A': 0.25152017429930257, 'B': 0.7484798257006975}
Classification: INCORRECT
----------------------------------------------------
Data point:     [4.08407045 2.70176968]
Label:          A
Assigned:       A
Probabilities:  {'A': 0.5539841970276697, 'B': 0.44601580297233046}
Classification: CORRECT
----------------------------------------------------
Data point:     [0.43982297 2.0106193 ]
Label:          A
Assigned:       A
```

```
Probabilities:  {'A': 0.6009235381067621, 'B': 0.3990764618932378}
Classification: CORRECT
---------------------------------------------------
Data point:     [1.31946891 2.51327412]
Label:          A
Assigned:       A
Probabilities:  {'A': 0.5054581668143066, 'B': 0.49454183318569334}
Classification: CORRECT
---------------------------------------------------
Data point:     [2.70176968 3.95840674]
Label:          A
Assigned:       A
Probabilities:  {'A': 0.5518900962793769, 'B': 0.4481099037206231}
Classification: CORRECT
---------------------------------------------------
Data point:     [4.77522083 0.62831853]
Label:          A
Assigned:       B
Probabilities:  {'A': 0.46916869950296314, 'B': 0.530831300497037}
Classification: INCORRECT
---------------------------------------------------
Data point:     [1.00530965 5.59203492]
Label:          A
Assigned:       A
Probabilities:  {'A': 0.6577284429728674, 'B': 0.34227155702713274}
Classification: CORRECT
---------------------------------------------------
Data point:     [5.40353936 3.58141563]
Label:          A
Assigned:       A
Probabilities:  {'A': 0.6478382963481608, 'B': 0.3521617036518392}
Classification: CORRECT
---------------------------------------------------
Data point:     [1.63362818 3.83274304]
Label:          A
Assigned:       B
Probabilities:  {'A': 0.43212555766573196, 'B': 0.5678744423342681}
Classification: INCORRECT
---------------------------------------------------
Data point:     [4.64955713 2.07345115]
Label:          A
Assigned:       B
Probabilities:  {'A': 0.1551929222958315, 'B': 0.8448070777041685}
Classification: INCORRECT
---------------------------------------------------
Data point:     [2.89026524 4.27256601]
Label:          A
Assigned:       A
```

```
Probabilities:  {'A': 0.5268972847186237, 'B': 0.47310271528137626}
Classification: CORRECT
----------------------------------------------------
Data point:     [0.         0.56548668]
Label:          A
Assigned:       A
Probabilities:  {'A': 0.8034794365521127, 'B': 0.1965205634478872}
Classification: CORRECT
----------------------------------------------------
Data point:     [0.         1.88495559]
Label:          A
Assigned:       A
Probabilities:  {'A': 0.5026788839750737, 'B': 0.49732111602492646}
Classification: CORRECT
----------------------------------------------------
Data point:     [1.50796447 3.95840674]
Label:          A
Assigned:       B
Probabilities:  {'A': 0.25152017429930257, 'B': 0.7484798257006975}
Classification: INCORRECT
----------------------------------------------------
Data point:     [5.34070751 5.71769863]
Label:          B
Assigned:       B
Probabilities:  {'A': 0.3658540001087979, 'B': 0.634145999891202}
Classification: CORRECT
----------------------------------------------------
Data point:     [0.62831853 4.1469023 ]
Label:          B
Assigned:       B
Probabilities:  {'A': 0.38405562829280543, 'B': 0.6159443717071945}
Classification: CORRECT
----------------------------------------------------
Data point:     [0.25132741 2.38761042]
Label:          B
Assigned:       A
Probabilities:  {'A': 0.6474229984661243, 'B': 0.3525770015338757}
Classification: INCORRECT
----------------------------------------------------
Data point:     [4.27256601 2.19911486]
Label:          B
Assigned:       B
Probabilities:  {'A': 0.48761079358867965, 'B': 0.5123892064113204}
Classification: CORRECT
----------------------------------------------------
Data point:     [0.9424778  2.26194671]
Label:          B
Assigned:       A
```

```
Probabilities:  {'A': 0.5562909902888116, 'B': 0.44370900971118843}
Classification: INCORRECT
----------------------------------------------------
Data point:     [5.84336234 3.0787608 ]
Label:          B
Assigned:       B
Probabilities:  {'A': 0.12427790976355005, 'B': 0.87572209023645}
Classification: CORRECT
----------------------------------------------------
Data point:     [3.39292007 3.95840674]
Label:          B
Assigned:       B
Probabilities:  {'A': 0.24114846994408362, 'B': 0.7588515300559164}
Classification: CORRECT
----------------------------------------------------
Data point:     [2.19911486 3.20442451]
Label:          B
Assigned:       B
Probabilities:  {'A': 0.36356059300504173, 'B': 0.6364394069949583}
Classification: CORRECT
----------------------------------------------------
Data point:     [2.57610598 0.06283185]
Label:          B
Assigned:       A
Probabilities:  {'A': 0.8497394684227537, 'B': 0.15026053157724636}
Classification: INCORRECT
----------------------------------------------------
Data point:     [2.136283   1.00530965]
Label:          B
Assigned:       A
Probabilities:  {'A': 0.5929869347646791, 'B': 0.4070130652353209}
Classification: INCORRECT
----------------------------------------------------
Data point:     [4.20973416 5.0893801 ]
Label:          B
Assigned:       B
Probabilities:  {'A': 0.420162513464851, 'B': 0.5798374865351489}
Classification: CORRECT
----------------------------------------------------
Data point:     [5.02654825 6.09468975]
Label:          B
Assigned:       B
Probabilities:  {'A': 0.4763291525984219, 'B': 0.5236708474015781}
Classification: CORRECT
----------------------------------------------------
Data point:     [0.87964594 5.0893801 ]
Label:          B
Assigned:       B
```

```
Probabilities:  {'A': 0.42186578782767326, 'B': 0.5781342121723267}
Classification: CORRECT
----------------------------------------------------
Data point:     [5.40353936 2.57610598]
Label:          B
Assigned:       B
Probabilities:  {'A': 0.40571516785301, 'B': 0.5942848321469899}
Classification: CORRECT
----------------------------------------------------
Data point:     [4.08407045 0.06283185]
Label:          B
Assigned:       B
Probabilities:  {'A': 0.3489928320932421, 'B': 0.6510071679067579}
Classification: CORRECT
----------------------------------------------------
Data point:     [2.51327412 3.26725636]
Label:          B
Assigned:       B
Probabilities:  {'A': 0.246178063398314, 'B': 0.7538219366016861}
Classification: CORRECT
----------------------------------------------------
Data point:     [0.75398224 1.38230077]
Label:          B
Assigned:       B
Probabilities:  {'A': 0.35485351758867345, 'B': 0.6451464824113265}
Classification: CORRECT
----------------------------------------------------
Data point:     [0.75398224 2.82743339]
Label:          B
Assigned:       A
Probabilities:  {'A': 0.6245224985670756, 'B': 0.3754775014329244}
Classification: INCORRECT
----------------------------------------------------
Data point:     [2.63893783 3.0787608 ]
Label:          B
Assigned:       B
Probabilities:  {'A': 0.1151236256138169, 'B': 0.8848763743861832}
Classification: CORRECT
----------------------------------------------------
Data point:     [4.08407045 4.77522083]
Label:          B
Assigned:       B
Probabilities:  {'A': 0.21330881979127886, 'B': 0.7866911802087211}
Classification: CORRECT

67.5 % of the test data was correctly classified!
```


A
B
6
5
4
3
2
1
0
0
1
2
3
4
5
6