MultiLayerNN.DecisionBoundary

June 10, 2021

[1]:
```
                                    # CAP 5615 2021 Summer, X. Zhu, June 10 2021
                                       # Multi Layer Neural Network Learning
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.utils import shuffle
```

[2]:
```
# create moon shaped dataset
from sklearn.datasets import make_moons, make_classification
moon =make_moons(noise=0.3, random_state=0)
features,labels=moon
print(features)
print(labels)
```

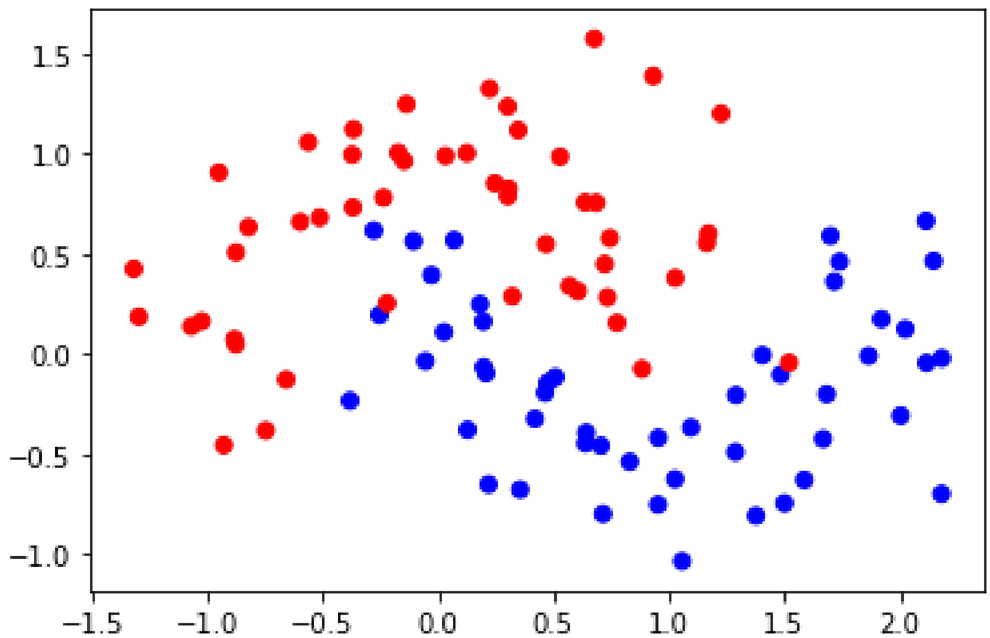
```
[[ 0.03159499  0.98698776]
 [ 2.11509784 -0.04624397]
 [ 0.88248972 -0.07575606]
 [-0.0551441  -0.03733246]
 [ 0.82954503 -0.53932149]
 [ 2.11285708  0.66208353]
 [ 0.5696927   0.33744136]
 [ 0.95217454 -0.75307471]
 [-0.02922115  0.39287336]
 [ 1.71579557  0.36069454]
 [-0.27883014  0.61420539]
 [ 2.00344509 -0.3091496 ]
 [ 1.50009616 -0.74636376]
 [ 1.22540835  1.19793017]
 [ 0.46730819  0.54703192]
 [-0.36653222  1.11971633]
 [ 0.30124459  1.23315697]
 [ 0.30430746  0.82373935]
 [-0.23712492  0.77876034]
 [ 1.16339832  0.55290238]
 [-0.59702417  0.6572361 ]
 [ 0.46356735 -0.1951543 ]
 [ 0.68451111  0.75287685]
```

```
[ 0.70515699 -0.45892444]
[-0.65805008 -0.12944211]
[-0.74662946 -0.3829632 ]
[ 0.60585226  0.31252842]
[ 2.18137168 -0.02291747]
[ 1.91980633  0.17247329]
[ 1.4834364  -0.10517023]
[ 0.47404723 -0.14575067]
[ 1.05614605 -1.03570207]
[ 1.86500732 -0.01107874]
[ 0.4194721  -0.32478101]
[ 0.06873258  0.56648467]
[-0.17332432  1.00215131]
[ 0.12412421  1.00062892]
[ 0.18121142  0.24717743]
[-0.25451559  0.19317272]
[ 1.02580668 -0.62574566]
[ 1.52002143 -0.04515581]
[ 0.64174037 -0.39369468]
[-0.87615589  0.0465662 ]
[-1.06964997  0.13666025]
[ 1.02658765  0.37782458]
[ 0.93131325  1.38517841]
[ 0.67680544  1.57189931]
[-0.36885733  0.72886601]
[-1.02465495  0.16190215]
[ 0.77455385  0.15375803]
[ 1.4045116  -0.00705701]
[-0.38147174 -0.23488747]
[ 0.72155224  0.44721658]
[-0.51346686  0.67869095]
[ 0.32118546  0.28668667]
[ 0.1953628   0.16085107]
[ 0.52824196  0.98300993]
[-0.2216539   0.25160139]
[ 0.22334676  1.32217183]
[-0.10704572  0.56178326]
[ 0.63651685  0.75444825]
[-0.37227848  0.99291317]
[ 0.20718083 -0.09767143]
[ 0.12733142 -0.3796549 ]
[ 0.71435231 -0.79994161]
[ 1.09487814 -0.36841845]
[-0.14814362  0.96158657]
[ 1.586188   -0.62984517]
[ 0.74444551  0.57661371]
[ 2.18011028 -0.69977751]
[ 0.24575594  0.8496383 ]
```

```
 [-0.95003581  0.90361699]
 [-0.88230758  0.07249044]
 [ 2.02297079  0.12325148]
 [ 2.14577321  0.46296362]
 [ 0.35536468 -0.67847989]
 [ 0.34665026  1.11570676]
 [ 1.7392373   0.45900352]
 [ 0.63856467 -0.44718443]
 [ 1.2876687  -0.4910366 ]
 [-0.13772607  1.2453262 ]
 [-0.56175303  1.05486051]
 [ 1.29003748 -0.20691405]
 [-0.87539365  0.50543423]
 [-0.92858249 -0.45631991]
 [ 0.02493632  0.10747958]
 [ 0.1972559  -0.06801668]
 [ 0.73346056  0.28161929]
 [ 1.68294434 -0.2020423 ]
 [ 0.50764124 -0.11731979]
 [ 1.66760217 -0.42485665]
 [-0.82172282  0.63141066]
 [ 0.30170903  0.78603534]
 [ 1.37671505 -0.80915107]
 [ 1.17037551  0.59840653]
 [ 1.69945309  0.58771967]
 [ 0.21862323 -0.65252119]
 [ 0.95291428 -0.41976564]
 [-1.31850034  0.42311235]
 [-1.29681764  0.18414709]]
[0 1 0 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 1 1 1 1 1 1 1 0 0
 1 1 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 0 0 1 1 1 1 0 1 0 1 0 0 0 1
 1 1 0 1 1 1 0 0 1 0 0 1 1 0 1 1 1 0 0 1 0 1 1 1 0 0]
```

[3]:
```python
colors=["red","blue"]
plt.scatter(features[:,0],features[:,1],color=[colors[idx] for idx in labels])
```

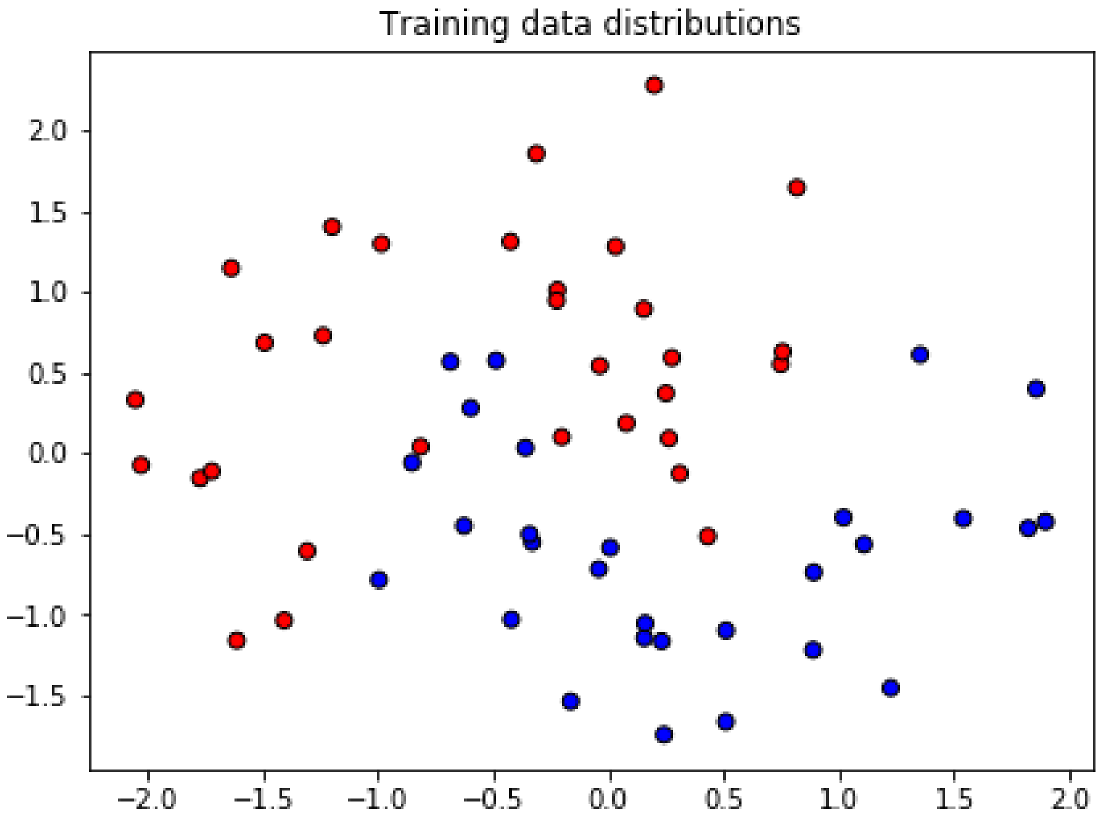
[3]:
```
<matplotlib.collections.PathCollection at 0x2d5ae7364c8>
```



[4]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
h = .02  # step size in the mesh
X, y=features,labels
X = StandardScaler().fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.4,␣
 ↪random_state=42)
```

[5]:
```
# just plot the dataset first
from matplotlib.colors import ListedColormap
plt.subplots(figsize = (15,5))
cm = plt.cm.RdBu
cm_bright = ListedColormap(['#FF0000', '#0000FF'])
ax = plt.subplot(1, 2, 1)
ax.set_title("Training data distributions")
# Plot the training points
ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm_bright,␣
 ↪edgecolors='k')
# Plot the testing points
#ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cm_bright, alpha=0.4,␣
 ↪edgecolors='k')
```

[5]:
```
<matplotlib.collections.PathCollection at 0x2d5ae728fc8>
```



[6]:
```
from sklearn.neural_network import MLPClassifier
clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=3,␣
 ↪random_state=12,activation='logistic',max_iter=500)
clf.fit(X_train, y_train)
```

[6]:
```
MLPClassifier(activation='logistic', alpha=1e-05, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=3, learning_rate='constant',
              learning_rate_init=0.001, max_iter=500, momentum=0.9,
              n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
              random_state=12, shuffle=True, solver='lbfgs', tol=0.0001,
              validation_fraction=0.1, verbose=False, warm_start=False)
```

[7]:
```
# neuro weights without considering bias
print(clf.coefs_)
# neron bias weights
print(clf.intercepts_)
```

```
[array([[ -21.32824873,   92.40872322,  -29.82416099],
       [  23.21072106, -133.01165313,   92.8777694 ]]), array([[-31.133009  ],
       [ 33.69456511],
```

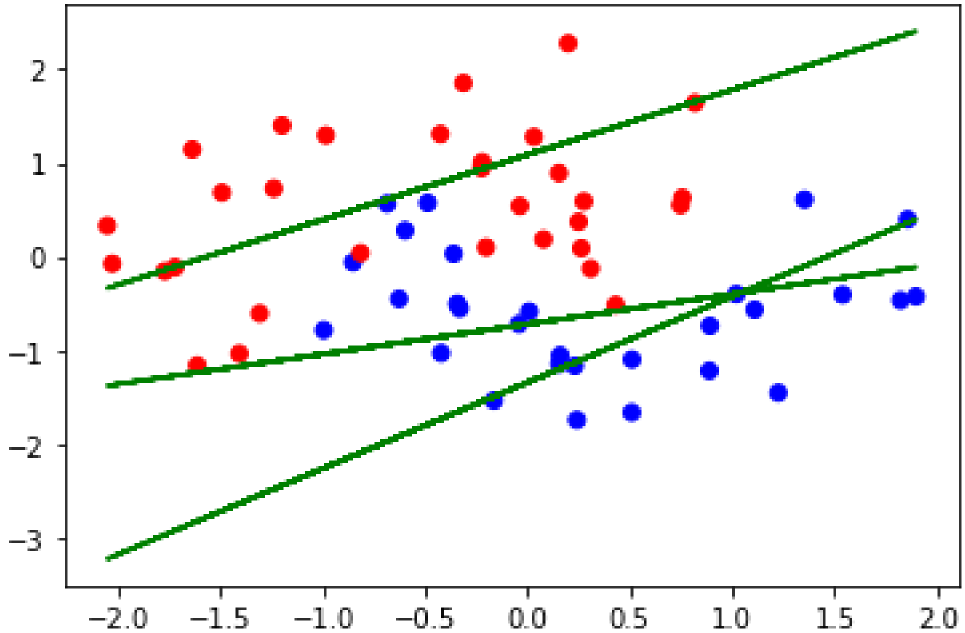
```
        [-26.25454745]])]
[array([ 31.20098212, 144.68848442,  66.93933631]), array([23.38176657])]
```

[9]:
```
# find slope and intercept of the line corresponnding to each hideen node (1st␣
↪hidden layer)
def findSlopeIntercept(coefs,bias):
    hidden_wts=coefs[0]
    bias_wts=bias[0]
    num=len(hidden_wts[0])
    slopIntercept=[]
    for i in range(num):
        w=[bias_wts[i],hidden_wts[0,i],hidden_wts[1,i]]
        slope=w[1]/w[2]*(-1)
        intercept=w[0]/w[2]*(-1)
        slopIntercept.append([slope,intercept])
    return(slopIntercept)
```

[10]:
```
slopeIntercept=findSlopeIntercept(clf.coefs_,clf.intercepts_)
slopeIntercept
```

[10]:
```
[[0.9188964305826433, -1.3442487221667776],
 [0.6947415586924526, 1.0877880322288032],
 [0.3211119429484934, -0.7207250641239307]]
```

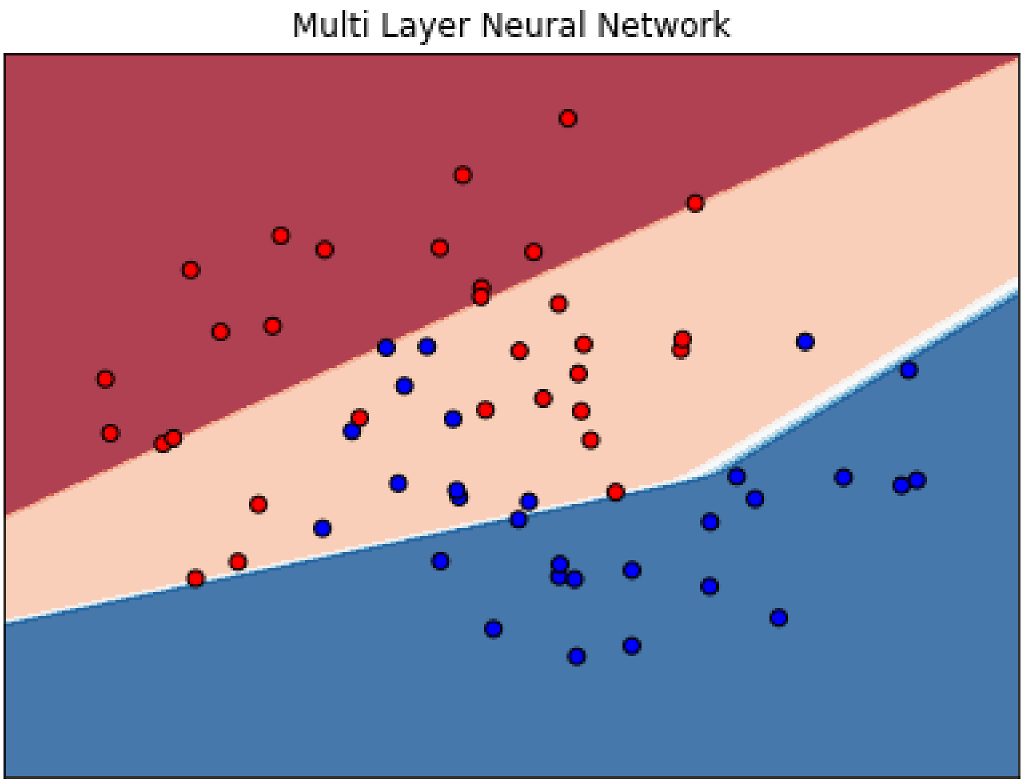
[11]:
```
colors=["red","blue"]
xvalues=X_train[:,0]
plt.scatter(X_train[:,0],X_train[:,1],color=[colors[idx] for idx in y_train])
num=len(slopeIntercept)
for i in range(num):
    yvalues=xvalues*slopeIntercept[i][0]+slopeIntercept[i][1]
    plt.plot(xvalues,yvalues,"g-")
```



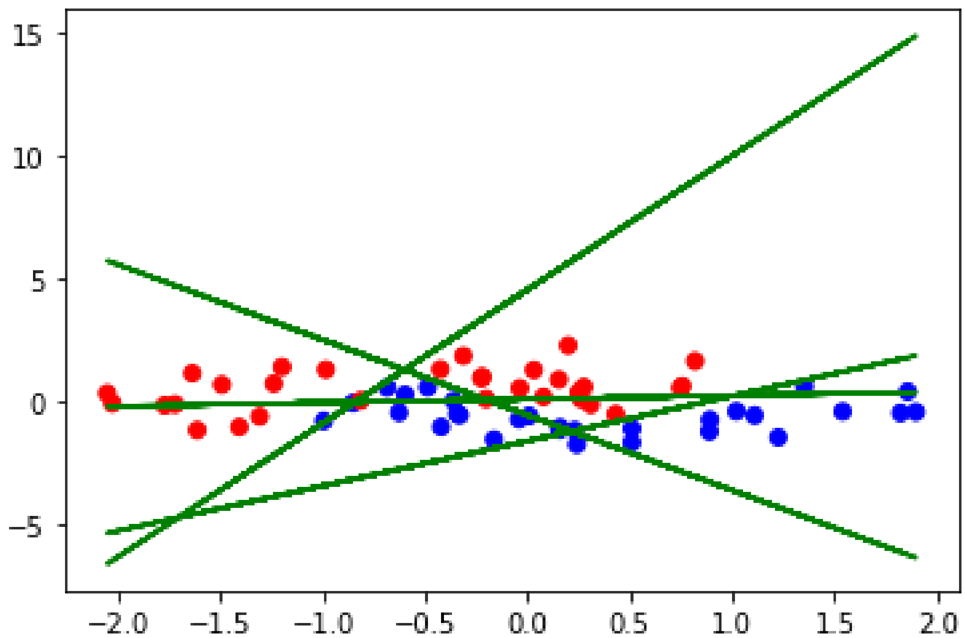
[12]:
```
# Plot the decision boundary. For that, we will assign a color to each
# point in the mesh [x_min, x_max]x[y_min, y_max].

x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))

plt.subplots(figsize = (15,5))
ax = plt.subplot(1, 2, 2)
if hasattr(clf, "decision_function"):
    Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
else:
    Z = clf.predict_proba(np.c_[xx.ravel(), yy.ravel()])[:, 1]

# Put the result into a color plot
Z = Z.reshape(xx.shape)
ax.contourf(xx, yy, Z, cmap=cm, alpha=.8)

# Plot the training points
ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm_bright,␣
 ↪edgecolors='k')

ax.set_xlim(xx.min(), xx.max())
ax.set_ylim(yy.min(), yy.max())
ax.set_xticks(())
ax.set_yticks(())
```

```
ax.set_title('Multi Layer Neural Network')
```

[12]: Text(0.5, 1.0, 'Multi Layer Neural Network')

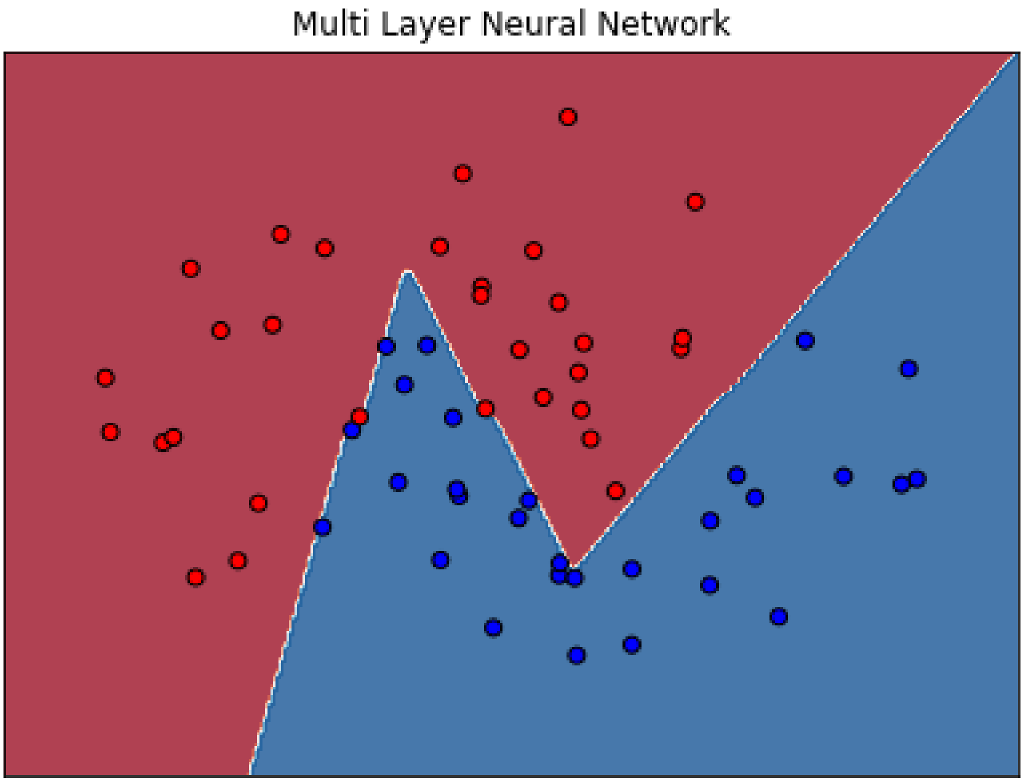


[13]:
```
# now increase to 4 hidden nodes
clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=4,␣
 ↪random_state=12,activation='logistic',max_iter=500)
clf.fit(X_train, y_train)
slopeIntercept=findSlopeIntercept(clf.coefs_,clf.intercepts_)
plt.scatter(X_train[:,0],X_train[:,1],color=[colors[idx] for idx in y_train])
num=len(slopeIntercept)
for i in range(num):
    yvalues=xvalues*slopeIntercept[i][0]+slopeIntercept[i][1]
    plt.plot(xvalues,yvalues,"g-")
```

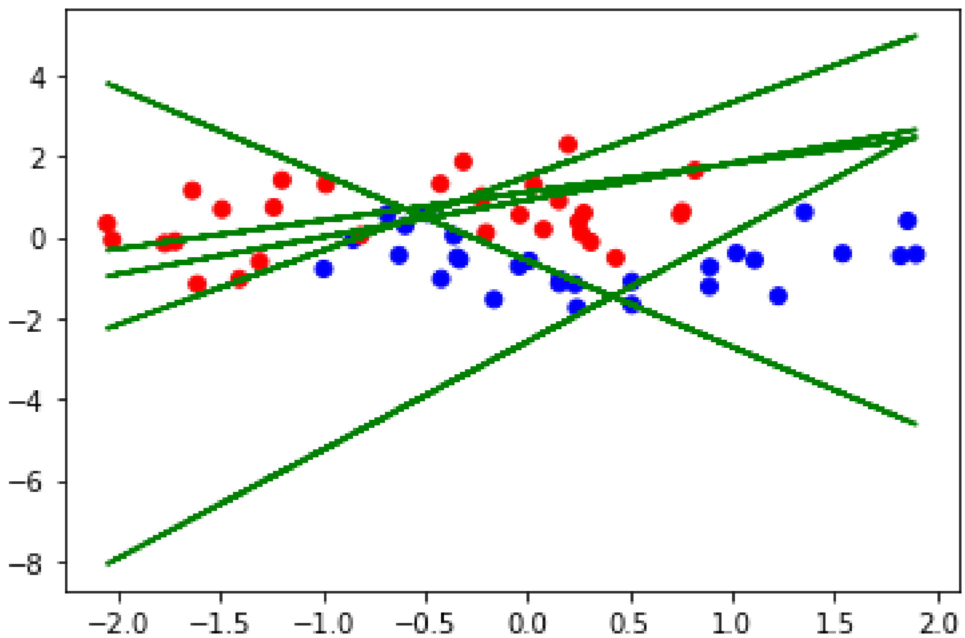


[14]:
```
# Plot the decision boundary. For that, we will assign a color to each
# point in the mesh [x_min, x_max]x[y_min, y_max].
plt.subplots(figsize = (15,5))
ax = plt.subplot(1, 2, 2)
if hasattr(clf, "decision_function"):
    Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
else:
    Z = clf.predict_proba(np.c_[xx.ravel(), yy.ravel()])[:, 1]

# Put the result into a color plot
Z = Z.reshape(xx.shape)
ax.contourf(xx, yy, Z, cmap=cm, alpha=.8)

# Plot the training points
ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm_bright,␣
 ↪edgecolors='k')

ax.set_xlim(xx.min(), xx.max())
ax.set_ylim(yy.min(), yy.max())
ax.set_xticks(())
ax.set_yticks(())
ax.set_title('Multi Layer Neural Network')
```

[14]: Text(0.5, 1.0, 'Multi Layer Neural Network')



[15]:
```
# now increase to 5 hidden nodes. As we add more hidden nodes, the model is␣
↪becoming more complex
# and therefore, have a higher risk of overfitting
clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=5,␣
↪random_state=12,activation='logistic',max_iter=500)
clf.fit(X_train, y_train)
slopeIntercept=findSlopeIntercept(clf.coefs_,clf.intercepts_)
plt.scatter(X_train[:,0],X_train[:,1],color=[colors[idx] for idx in y_train])
num=len(slopeIntercept)
for i in range(num):
    yvalues=xvalues*slopeIntercept[i][0]+slopeIntercept[i][1]
    plt.plot(xvalues,yvalues,"g-")
```



[16]:
```
# Plot the decision boundary. For that, we will assign a color to each
# point in the mesh [x_min, x_max]x[y_min, y_max].
plt.subplots(figsize = (15,5))
ax = plt.subplot(1, 2, 2)
if hasattr(clf, "decision_function"):
    Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
else:
    Z = clf.predict_proba(np.c_[xx.ravel(), yy.ravel()])[:, 1]

# Put the result into a color plot
Z = Z.reshape(xx.shape)
ax.contourf(xx, yy, Z, cmap=cm, alpha=.8)

# Plot the training points
ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm_bright,␣
 ↪edgecolors='k')

ax.set_xlim(xx.min(), xx.max())
ax.set_ylim(yy.min(), yy.max())
ax.set_xticks(())
ax.set_yticks(())
ax.set_title('Multi Layer Neural Network')
```

[16]: Text(0.5, 1.0, 'Multi Layer Neural Network')

Multi Layer Neural Network