schulzd_Lab4

October 6, 2020

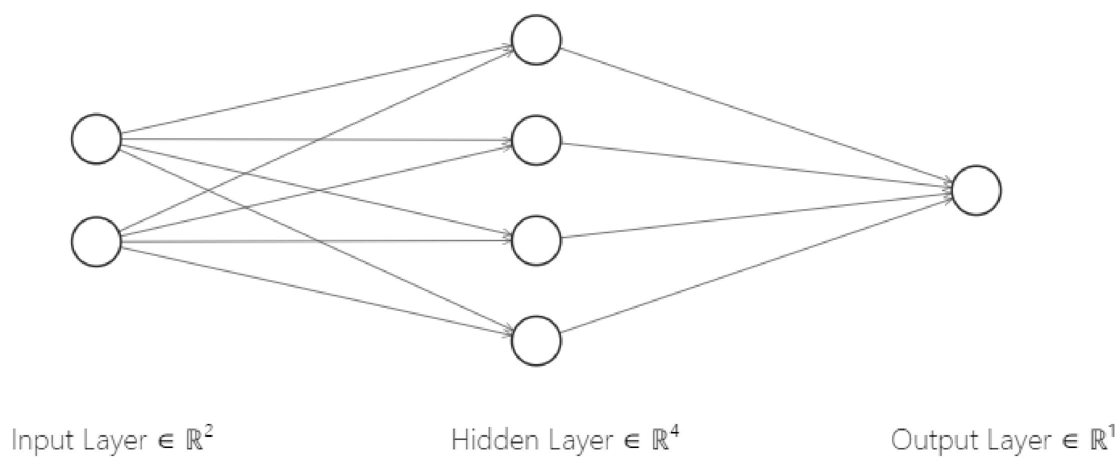
1 Lab 4 - Feedforward NN

David Schulz

1.1 Reflection Questions

1.1.1 Problem 1

1. What do the parameters to the MLPClassifier class mean?
   - hidden_layer_sizes specifies the number of neurons in each hidden layer (4 neurons), max_iter is the maximum number of iterations (1000), and solver is the solver for weight optimization ('lbfgs' is an optimizer in the family of quasi-Newton methods).
2. Draw the network.



3. What activation functions are used for each node?
   - ReLU

1.1.2 Problem 2

1. What are the dimensions of mlp_petals.coefs_[0] and mlp_petals.intercepts_[0]? Where do those dimensions come from?
   - mlp_petals.coefs_[0] is 2 x 4. These are the 2 weights for each neuron.
   - mlp_petals.intercepts_[0] is 1 x 4. These are the biases for each neuron.

2. What are the dimensions of mlp__petals__models? What do the dimensions correspond to?
    - mlp__petals__models is 4 x 3. Each row is a neuron and each column is the two weights and bias.
3. Which combination of planes can you use to separate setosa vs the rest, versicolor vs the rest, and viriginica vs the rest with the petal features? Create a table of planes versus the three class comparisons. Indicate in the table which planes are useful as decision boundaries for each particular comparison. (Planes may be used across multiple comparisons.)
    - A

1.1.3 Problem 3

1. The plane equation tells us which side of the plane a point is on (0 on the plane, < 0 one side, and > 0 other side). How does ReLU function modify the output and impact the interpretation?
    - The ReLU function makes it so that the output is 0 whenever the input is $<= 0$.
2. Recreate the table from problem 2.3 but using the heatmaps / contour plots of the model outputs with the ReLU activation layer.
    - A
3. How does the ReLU activation layer make it easier or more difficult to use the decision boundaries?
    - It makes it easier because it only needs to worry about one side of the decision boundary.

1.1.4 Problem 4

1. How do the confusion matrices and accuracies of the two models compare? Did the transformed features produce a more accurate model?
    - After running the tests multiple times, it didn't seem to make much of a difference. The chance of the transformed features making a more accurate model was 50/50. Even when the accuracy was better, it was only by 0.05 at most.

1.2 1. Train Multilayer Perceptron (MLP) Models on Petal and Sepal Features

[1]:
```
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier

iris = datasets.load_iris()

scaled_X = StandardScaler().fit_transform(iris.data)
mlp_petals = MLPClassifier(hidden_layer_sizes=(4,), max_iter=1000,␣
 ↪solver="lbfgs")
mlp_petals.fit(scaled_X[:, 2:], iris.target)

mlp_sepals = MLPClassifier(hidden_layer_sizes=(4,), max_iter=1000,␣
 ↪solver="lbfgs")
```

```python
mlp_sepals.fit(scaled_X[:, :2], iris.target)
```

[1]:
```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(4,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=1000,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='lbfgs',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
```

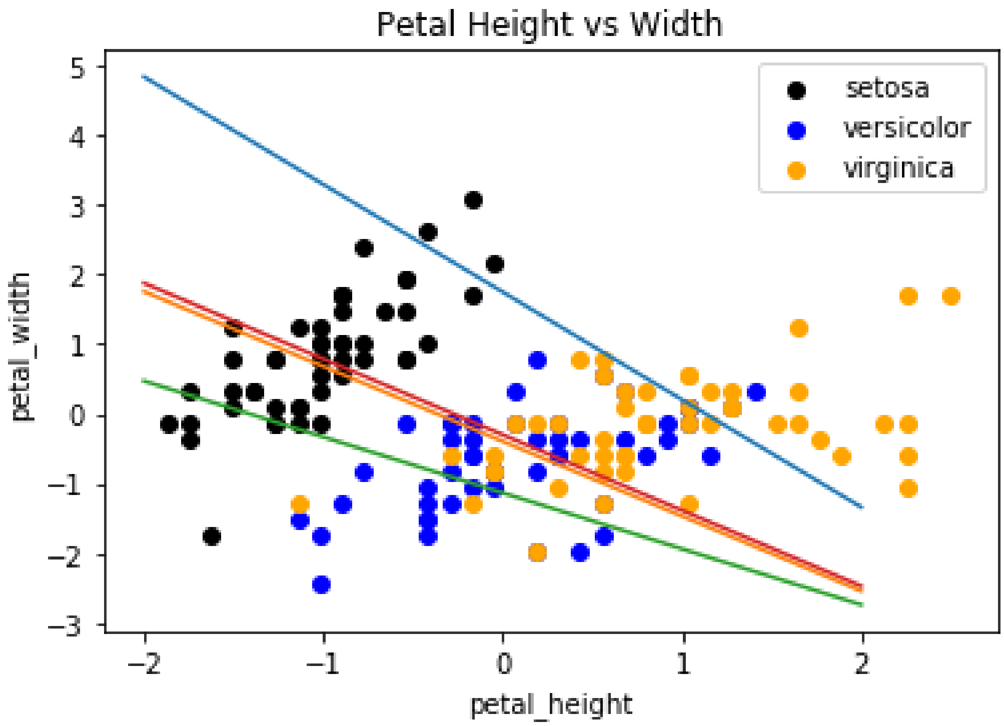
1.3 2. Visualize Planes Learned by Individual Neurons

[2]:
```python
import numpy as np

mlp_petals_models = np.vstack([mlp_petals.intercepts_[0], mlp_petals.
 ↪coefs_[0]]).T
mlp_sepals_models = np.vstack([mlp_sepals.intercepts_[0], mlp_sepals.
 ↪coefs_[0]]).T
```

x2 = (-B0 - B1x1) / B2

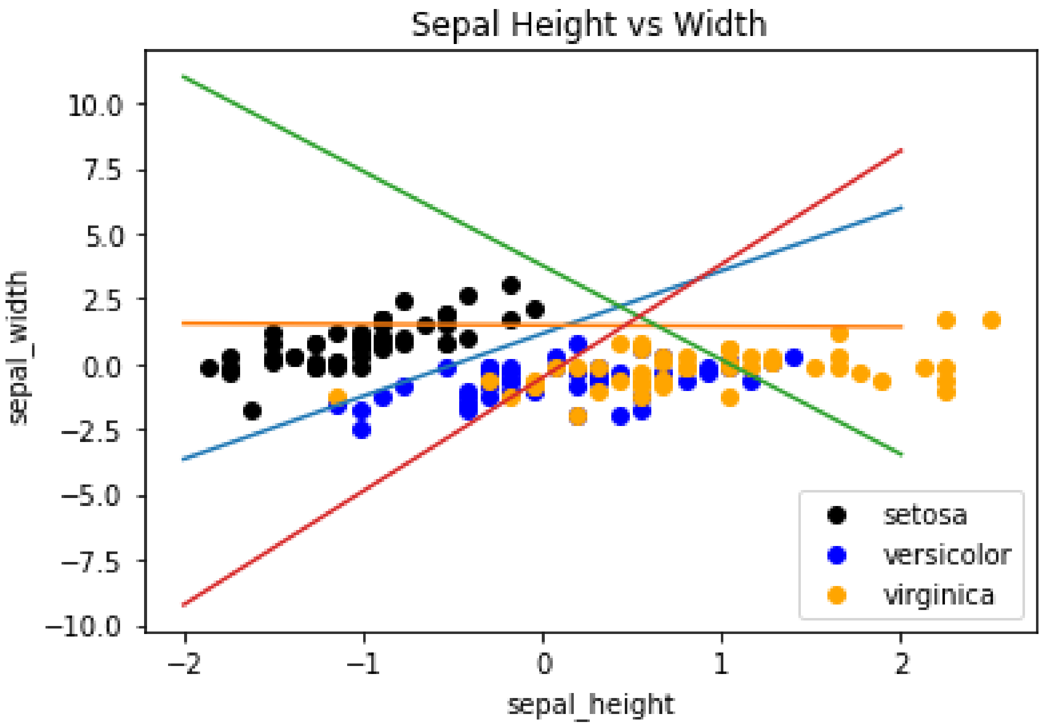
[3]:
```python
import matplotlib.pyplot as plt

x = np.linspace(-2, 2, num=100)

a = -mlp_petals_models[:, 1] / mlp_petals_models[:, 2]
b = mlp_petals_models[:, 0] / mlp_petals_models[:, 2]

for i in range(len(a)):
    plt.plot(x, (a[i] * x) - b[i])

class_0 = scaled_X[np.where(iris.target == 0)]
class_1 = scaled_X[np.where(iris.target == 1)]
class_2 = scaled_X[np.where(iris.target == 2)]

plt.scatter(class_0[:, 0], class_0[:, 1], c='black', label='setosa')
plt.scatter(class_1[:, 0], class_1[:, 1], c='blue', label='versicolor')
plt.scatter(class_2[:, 0], class_2[:, 1], c='orange', label='virginica')

plt.title('Petal Height vs Width')
plt.xlabel('petal_height')
plt.ylabel('petal_width')
plt.legend()
```

[3]:
```
<matplotlib.legend.Legend at 0x7f3fec131c90>
```



[4]:
```
a = -mlp_sepals_models[:, 1] / mlp_sepals_models[:, 2]
b = mlp_sepals_models[:, 0] / mlp_sepals_models[:, 2]

for i in range(len(a)):
    plt.plot(x, (a[i] * x) - b[i])

plt.scatter(class_0[:, 0], class_0[:, 1], c='black', label='setosa')
plt.scatter(class_1[:, 0], class_1[:, 1], c='blue', label='versicolor')
plt.scatter(class_2[:, 0], class_2[:, 1], c='orange', label='virginica')

plt.title('Sepal Height vs Width')
plt.xlabel('sepal_height')
plt.ylabel('sepal_width')
plt.legend()
```

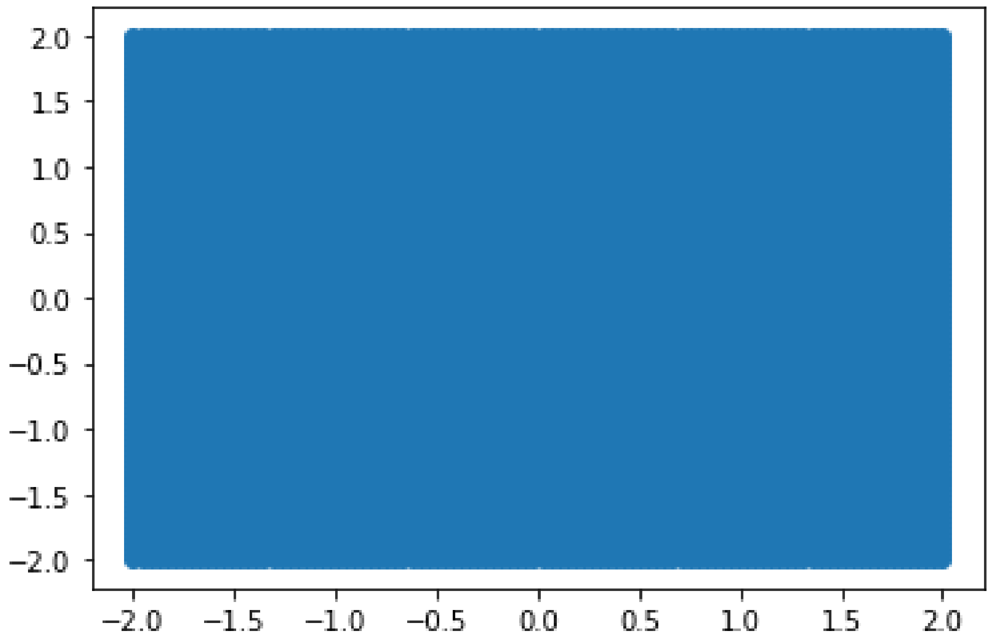
[4]: <matplotlib.legend.Legend at 0x7f3fec0fe850>



1.4 3. Visualize Decision Boundaries Resulting from Planes and ReLU Activation Function

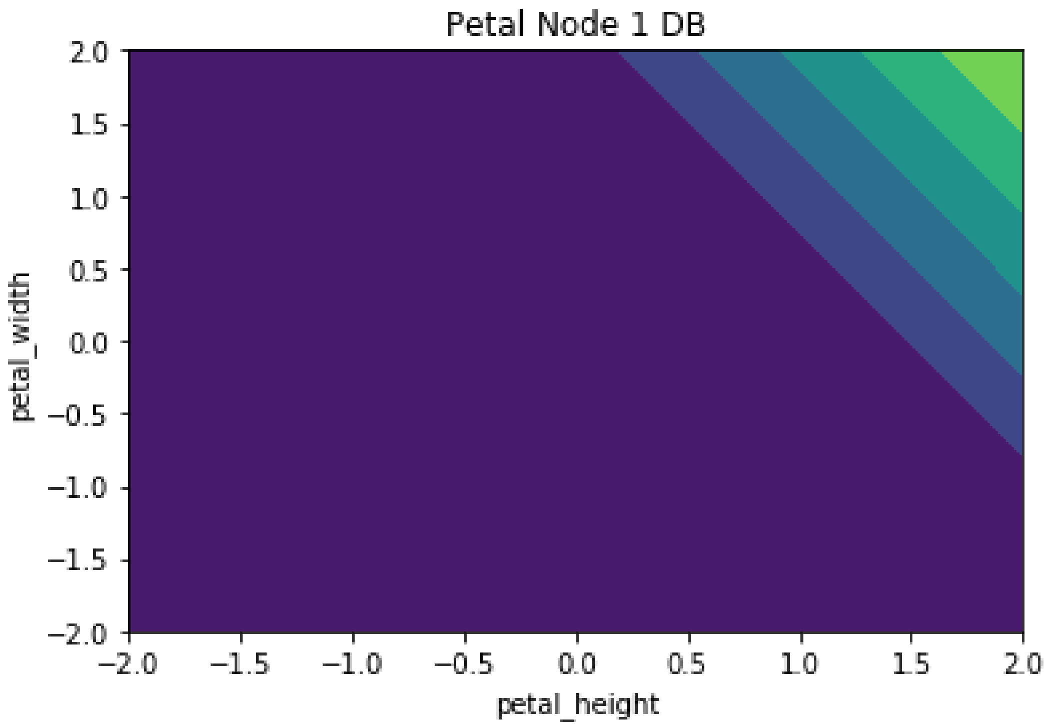
[5]:
```
xx, yy = np.meshgrid(x, x)
plt.scatter(xx, yy)
```

[5]: <matplotlib.collections.PathCollection at 0x7f3febfbd8d0>



[6]:
```
from neurons import Input, Neuron, HStack
from seaborn import heatmap

input = Input()
p_layer = Neuron([input], mlp_petals_models[0, :])

points = np.hstack((xx.reshape((-1, 1)), yy.reshape((-1, 1))))
pred = p_layer.predict(points)

plt.title('Petal Node 1 DB')
plt.xlabel('petal_height')
plt.ylabel('petal_width')
plt.contourf(xx, yy, pred.reshape((100, 100)))
```

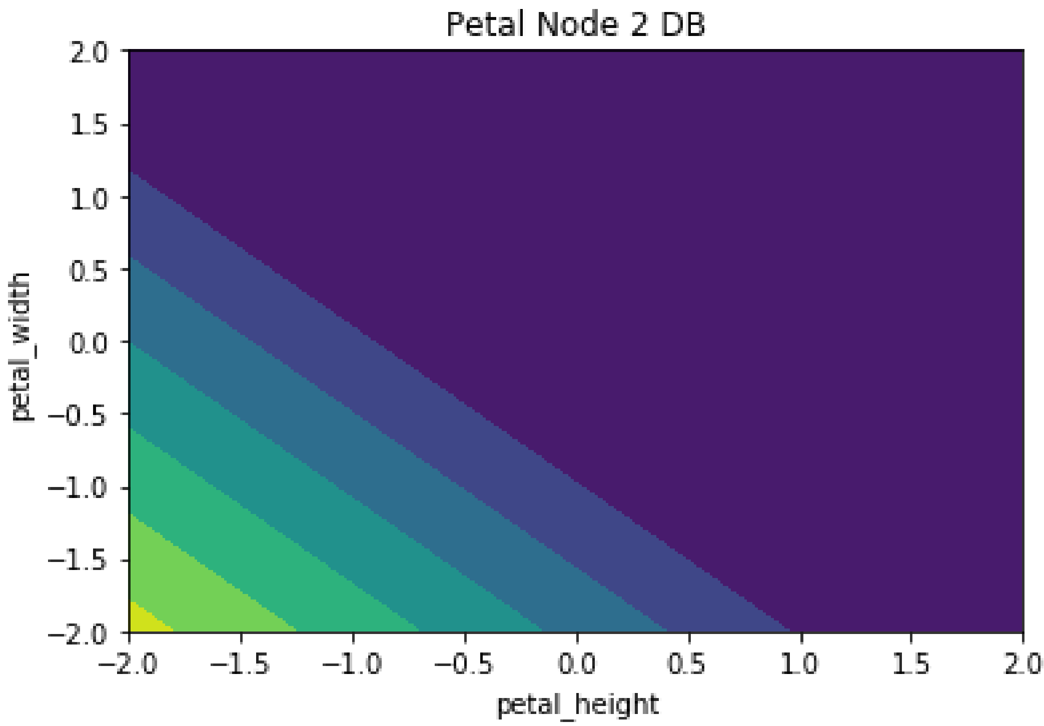
[6]:
```
<matplotlib.contour.QuadContourSet at 0x7f3fe4268b90>
```



[7]:

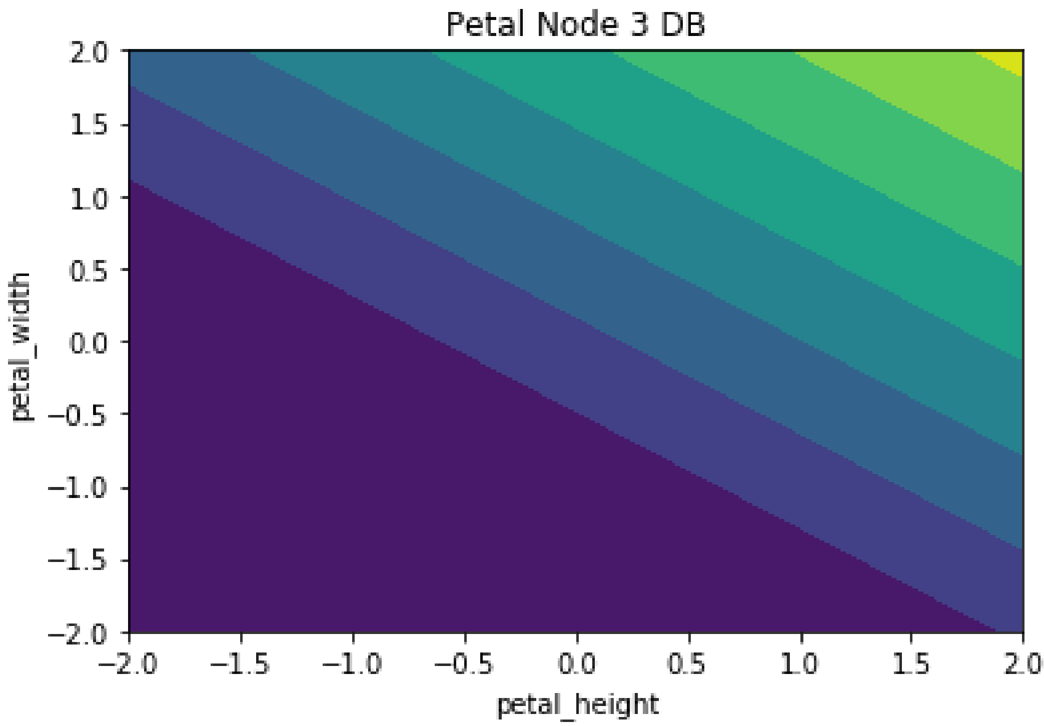
```
input = Input()
p_layer = Neuron([input], mlp_petals_models[1, :])

points = np.hstack((xx.reshape((-1, 1)), yy.reshape((-1, 1))))
pred = p_layer.predict(points)

plt.title('Petal Node 2 DB')
plt.xlabel('petal_height')
plt.ylabel('petal_width')
plt.contourf(xx, yy, pred.reshape((100, 100)))
```

[7]: <matplotlib.contour.QuadContourSet at 0x7f3fe42857d0>



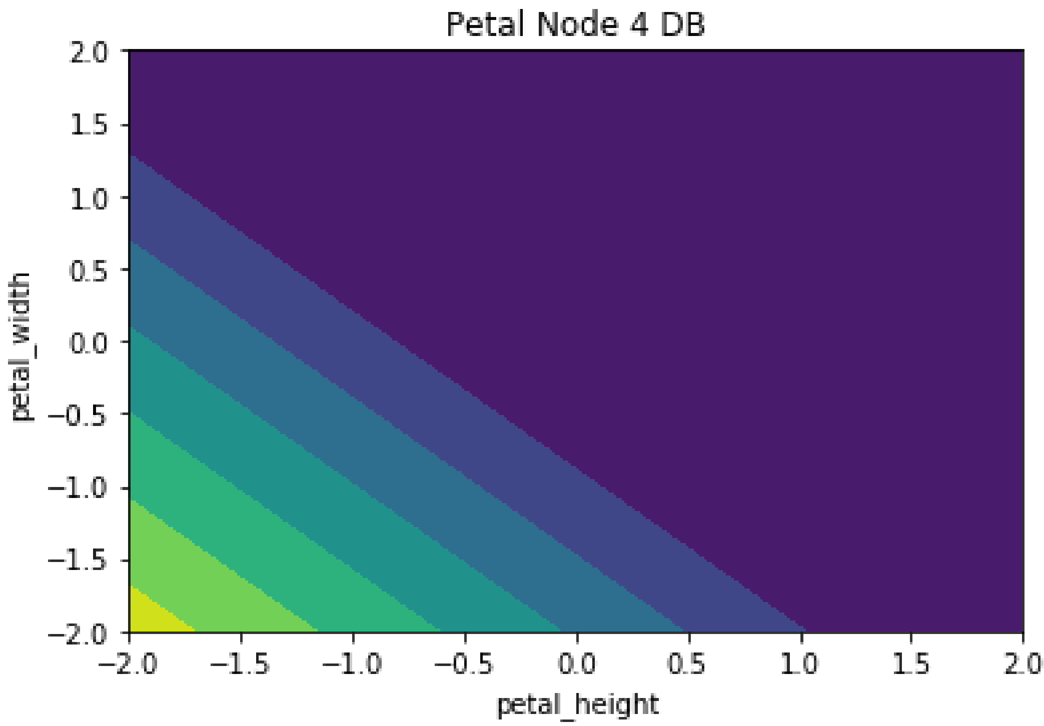
[8]:
```python
input = Input()
p_layer = Neuron([input], mlp_petals_models[2, :])

points = np.hstack((xx.reshape((-1, 1)), yy.reshape((-1, 1))))
pred = p_layer.predict(points)

plt.title('Petal Node 3 DB')
plt.xlabel('petal_height')
plt.ylabel('petal_width')
plt.contourf(xx, yy, pred.reshape((100, 100)))
```

[8]:
```
<matplotlib.contour.QuadContourSet at 0x7f3fe41fd550>
```



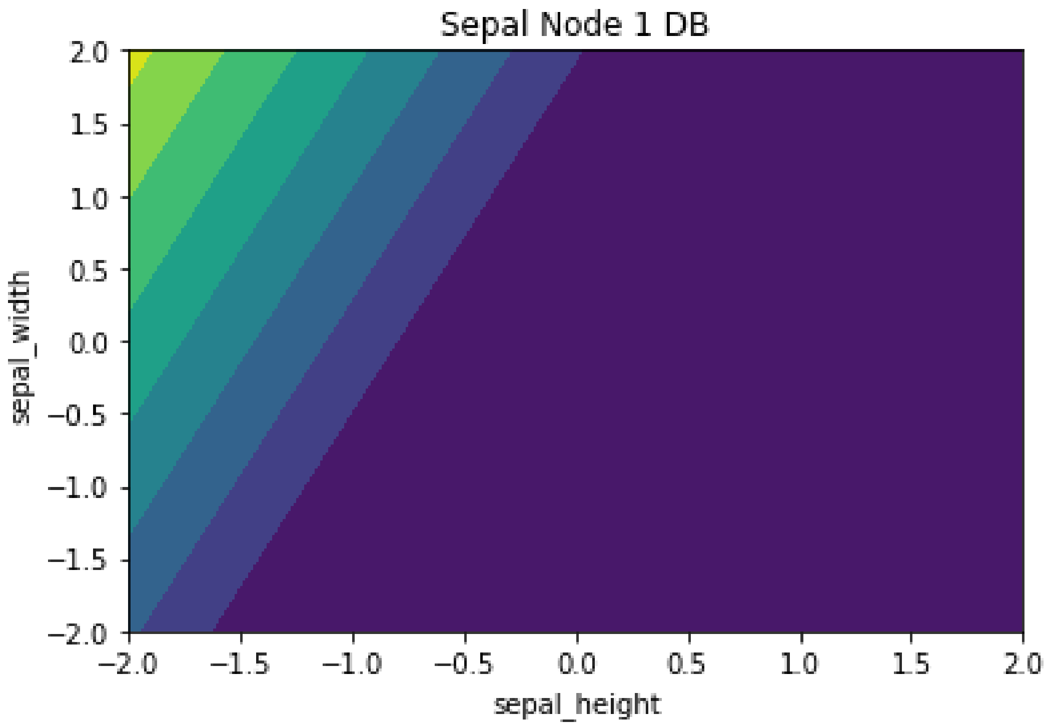
[9]:
```
input = Input()
p_layer = Neuron([input], mlp_petals_models[3, :])

points = np.hstack((xx.reshape((-1, 1)), yy.reshape((-1, 1))))
pred = p_layer.predict(points)

plt.title('Petal Node 4 DB')
plt.xlabel('petal_height')
plt.ylabel('petal_width')
plt.contourf(xx, yy, pred.reshape((100, 100)))
```

[9]: <matplotlib.contour.QuadContourSet at 0x7f3fe41853d0>



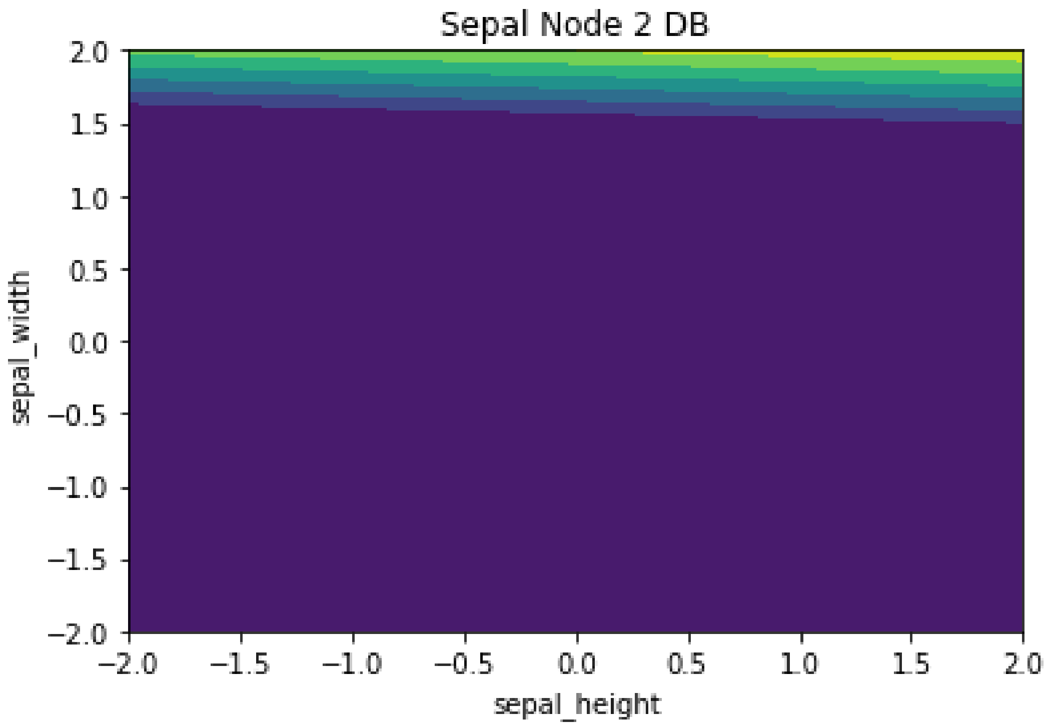
[10]:
```
input = Input()
p_layer = Neuron([input], mlp_sepals_models[0, :])

points = np.hstack((xx.reshape((-1, 1)), yy.reshape((-1, 1))))
pred = p_layer.predict(points)

plt.title('Sepal Node 1 DB')
plt.xlabel('sepal_height')
plt.ylabel('sepal_width')
plt.contourf(xx, yy, pred.reshape((100, 100)))
```

[10]:
```
<matplotlib.contour.QuadContourSet at 0x7f3fe41085d0>
```



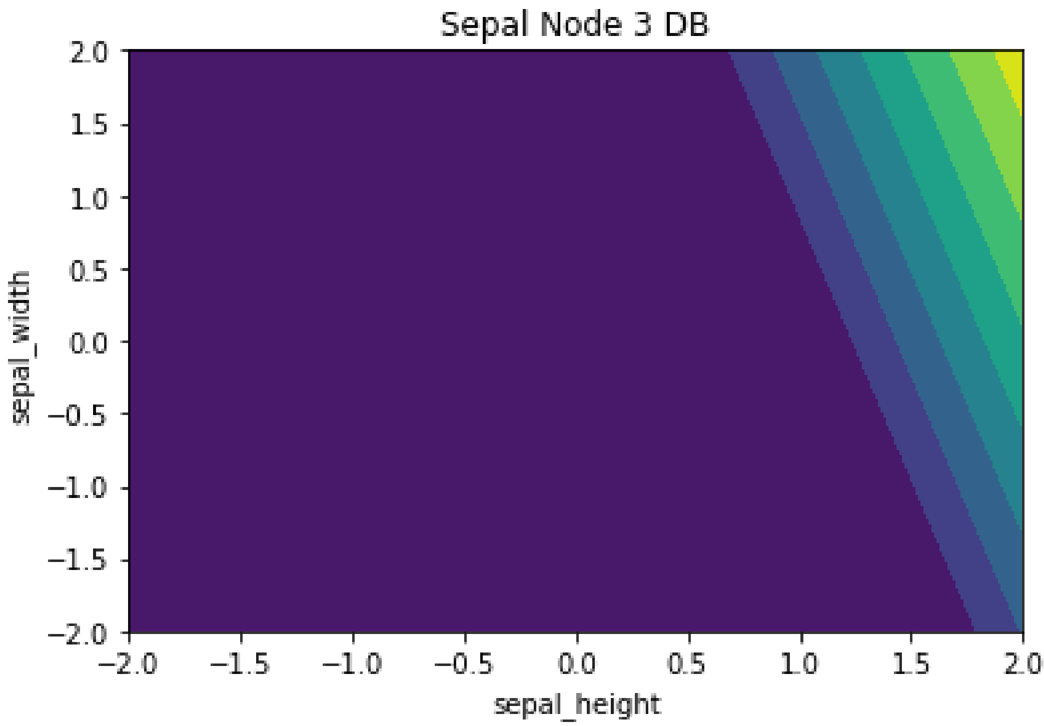
[11]:
```
input = Input()
p_layer = Neuron([input], mlp_sepals_models[1, :])

points = np.hstack((xx.reshape((-1, 1)), yy.reshape((-1, 1))))
pred = p_layer.predict(points)

plt.title('Sepal Node 2 DB')
plt.xlabel('sepal_height')
plt.ylabel('sepal_width')
plt.contourf(xx, yy, pred.reshape((100, 100)))
```

[11]:
```
<matplotlib.contour.QuadContourSet at 0x7f3fdffd18d0>
```



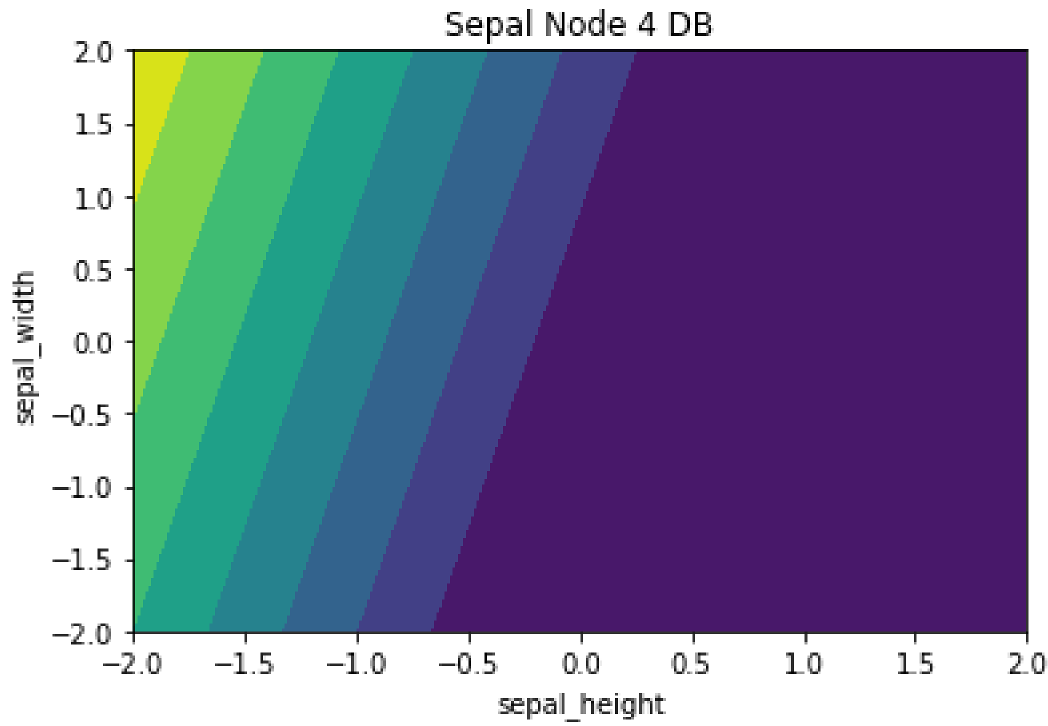
[12]:
```python
input = Input()
p_layer = Neuron([input], mlp_sepals_models[2, :])

points = np.hstack((xx.reshape((-1, 1)), yy.reshape((-1, 1))))
pred = p_layer.predict(points)

plt.title('Sepal Node 3 DB')
plt.xlabel('sepal_height')
plt.ylabel('sepal_width')
plt.contourf(xx, yy, pred.reshape((100, 100)))
```

[12]: <matplotlib.contour.QuadContourSet at 0x7f3fdfff5a90>



[13]:
```
input = Input()
p_layer = Neuron([input], mlp_sepals_models[3, :])

points = np.hstack((xx.reshape((-1, 1)), yy.reshape((-1, 1))))
pred = p_layer.predict(points)

plt.title('Sepal Node 4 DB')
plt.xlabel('sepal_height')
plt.ylabel('sepal_width')
plt.contourf(xx, yy, pred.reshape((100, 100)))
```

[13]: <matplotlib.contour.QuadContourSet at 0x7f3fdff014d0>



1.5 4. Train Logistic Regression Models on Transformed and Original Features

[14]:
```
input = Input()
p_layer_1 = Neuron([input], mlp_petals_models[0, :])
p_layer_2 = Neuron([input], mlp_petals_models[1, :])
p_layer_3 = Neuron([input], mlp_petals_models[2, :])
p_layer_4 = Neuron([input], mlp_petals_models[3, :])
stacked = HStack([p_layer_1, p_layer_2, p_layer_3, p_layer_4])
transformed_petals_X = stacked.predict(scaled_X[:, 2:])

input = Input()
p_layer_1 = Neuron([input], mlp_sepals_models[0, :])
p_layer_2 = Neuron([input], mlp_sepals_models[1, :])
p_layer_3 = Neuron([input], mlp_sepals_models[2, :])
p_layer_4 = Neuron([input], mlp_sepals_models[3, :])
stacked = HStack([p_layer_1, p_layer_2, p_layer_3, p_layer_4])
transformed_sepals_X = stacked.predict(scaled_X[:, 2:])

combined = np.hstack((transformed_petals_X, transformed_sepals_X))
```

[15]:
```
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import accuracy_score

X_train, X_test, y_train, y_test = train_test_split(scaled_X, iris.target,␣
↪test_size=0.33)

sgd = SGDClassifier(loss="log")
sgd.fit(X_train, y_train)
y_pred = sgd.predict(X_test)
accuracy_score(y_test, y_pred)
```

[15]: 0.94

[16]:
```
X_train, X_test, y_train, y_test = train_test_split(combined, iris.target,␣
↪test_size=0.33)

sgd = SGDClassifier(loss="log")
sgd.fit(X_train, y_train)
y_pred = sgd.predict(X_test)
accuracy_score(y_test, y_pred)
```

[16]: 0.9