In [502]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report,confusion_matrix
import warnings
warnings.filterwarnings('ignore')
```

In [503]:
```
df = pd.read_csv('iris.csv')
```

In [504]:
```
df.head()
```

Out[504]:

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [505]:
```
df.Species.unique()
```

Out[505]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [506]:
```
# For Accuracy testing
X1 = df[['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']]
y1 = df[['Species']]
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size =
```

In [507]:
```
y1_test.loc[y1_test["Species"]=="Iris-setosa","Species"]='Setosa'
y1_test.loc[y1_test["Species"]=="Iris-versicolor","Species"]='Versicolor'
y1_test.loc[y1_test["Species"]=="Iris-virginica","Species"]='Virginica'
```

In [508]:
```
y1_test.head()
```

Out[508]:

	Species
114	Virginica
62	Versicolor
33	Setosa
107	Virginica
7	Setosa

In [509]:
```
# Creating dummy variables for 3 unique category
df = pd.get_dummies(df, columns=['Species'],prefix='',prefix_sep='')
```

In [510]:
```
# df = pd.concat([dum,tmpdf], axis=1)
```

In [511]:
```
# df = df[['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width','Speci
```

In [512]:
```
# df.rename(columns={'Iris-setosa':'setosa','Iris-versicolor':'versicolor',
```

In [513]:
```
# Encoding
df.loc[df["Species"]=="Iris-setosa","Species"]=0
df.loc[df["Species"]=="Iris-versicolor","Species"]=1
df.loc[df["Species"]=="Iris-virginica","Species"]=2
```

In [514]:
```
# df = df[['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width','speci
```

In [515]:
```
df.rename(columns={'Iris-setosa':'setosa','Iris-versicolor':'versicolor','I
```

In [516]:
```
df.head()
```

Out[516]:

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	setosa	versicolor	virginica	Species
0	5.1	3.5	1.4	0.2	1	0	0	0
1	4.9	3.0	1.4	0.2	1	0	0	0
2	4.7	3.2	1.3	0.2	1	0	0	0
3	4.6	3.1	1.5	0.2	1	0	0	0
4	5.0	3.6	1.4	0.2	1	0	0	0

In [517]:
```
colsX = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']
colsy = ['setosa','virginica','versicolor']
X = df[colsX]
y = df[colsy]
```

In [ ]:

In [ ]:

In [520]:
```
# data_train = df[:105]
# data_test = df[105:150]
```

In [521]:
```
# data_train.head()
```

In [522]:
```
# data_test.head()
```

In [523]:
```
# Splitting data into training and testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,
```

In [524]:
```
X_train.head()
```

Out[524]:

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width
60	5.0	2.0	3.5	1.0
116	6.5	3.0	5.5	1.8
144	6.7	3.3	5.7	2.5
119	6.0	2.2	5.0	1.5
108	6.7	2.5	5.8	1.8

In [525]:
```
y_test.head()
```

Out[525]:

	setosa	virginica	versicolor
114	0	1	0
62	0	0	1
33	1	0	0
107	0	1	0
7	1	0	0

In [526]:
```
print(X_train.shape,y_train.shape,X_test.shape,y_test.shape)
```

```
(105, 4) (105, 3) (45, 4) (45, 3)
```

In [527]:
```
def sigmond(d):
    return(1/(1+np.exp(-d)))
```

In [528]:
```
def sigdiff(d):
    return (sigmond(d) * (1 - sigmond(d)))
```

In [529]:
```
def forward(d,w1,w2):
    z = np.dot(d,w1)
    z2 = sigmond(z)
    z3 = np.dot(z2,w2)
    o = sigmond(z3)

    return z,z2,z3,o
```

In [530]:
```
def backword(X,y,z,z2,z3,o,w1,w2):
    erro = np.subtract(o,y)
    deltao = erro * sigdiff(o)

    errz2 = np.dot(deltao,w2.T)
    deltaz2 = errz2 * sigdiff(z2)

    w1b = np.dot(X.T,deltaz2)
    w2b = np.dot(z2.T,deltao)

    return w1b,w2b
```

In [531]:
```
#  Workinggggg
# # Performing NN

# w1 = np.random.rand(4,7)
# w2 = np.random.rand(7,3)
# lr = 0.01
# loss = []

# for i in range(0,1000):
#     z,z2,z3,o = forward(data_train[colsX],w1,w2)
# #     o.rename(columns={0:'Setosa', 1:'Virginica', 2:'Versicolor'})
#     y = data_train[colsy]
#     loss_1 = np.sum((-y * np.log(o)).sum(axis = 1)/len(data_train))
#     loss.append(loss_1)
#     w1back, w2back = backword(data_train[colsX],y,z,z2,z3,o,w1,w2)

#     w1 = w1 - (lr*w1back)
#     w2 = w2 - (lr*w2back)
```

In [532]:
```
# Performing NN

w1 = np.random.rand(4,7)
w2 = np.random.rand(7,3)
lr = 0.01
loss = []
itr = []

for i in range(0,5000):
    itr.append(i)
    z,z2,z3,o = forward(X_train,w1,w2)
    loss_1 = np.sum((-y_train * np.log(o)).sum(axis = 1)/len(y_train))
    loss.append(loss_1)
    w1back, w2back = backword(X_train,y_train,z,z2,z3,o,w1,w2)

    w1 = w1 - (lr*w1back)
    w2 = w2 - (lr*w2back)
```

In [533]:
```
# # rough shell
#
```

In [534]:
```
z,z2,z3,o = forward(X_test,w1,w2)
o = pd.DataFrame(o, columns = ['Setosa','Virginica','Versicolor'])
o['Maximum'] = o[['Setosa','Virginica','Versicolor']].idxmax(axis=1)
```

In [554]:
```
loss = np.sort(loss)
```

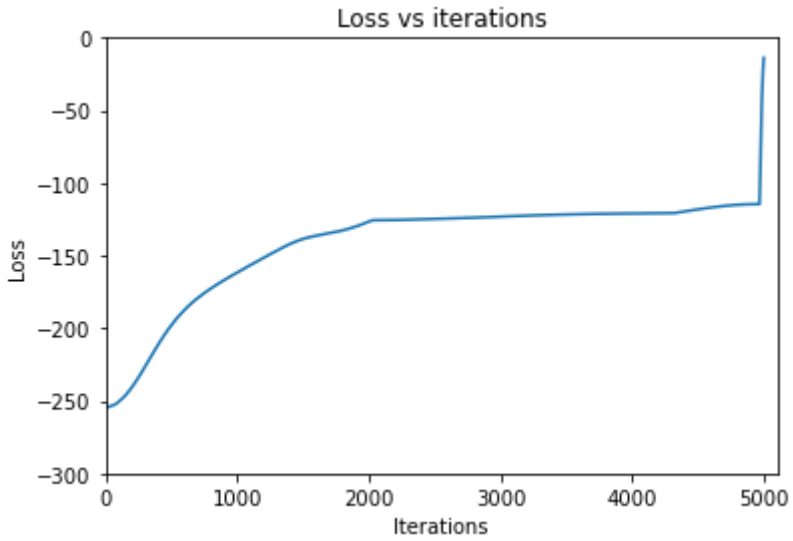
In [555]:
```
loss
```

Out[555]:
```
array([-254.39329423, -254.39208006, -254.39140103, ...,  -16.27992171,
        -14.97544645,  -13.7655333 ])
```

In [556]:
```
# Loss converges to 0
sns.lineplot(itr,loss)
plt.xlabel('Iterations')
plt.ylabel('Loss')
plt.title('Loss vs iterations')

plt.xlim(0,max(itr)+100)
plt.ylim(-300,0)
```

Out[556]:
```
(-300.0, 0.0)
```



In [557]:
```
o.head()
```

Out[557]:

	Setosa	Virginica	Versicolor	Maximum
0	0.005568	0.558609	0.433873	Virginica
1	0.006190	0.555549	0.438820	Virginica
2	0.983326	0.006021	0.024983	Setosa
3	0.005485	0.558994	0.433120	Virginica
4	0.979650	0.006789	0.026719	Setosa

In [561]:
```
# Acc
acc = accuracy_score(y_true = y1_test['Species'],y_pred =o['Maximum']) *100
print("Accuracy: ", acc, "%")
```

```
Accuracy:  60.0 %
```

In [560]:
```
print(classification_report(y1_test['Species'],o['Maximum']))
print(confusion_matrix(y1_test['Species'],o['Maximum']))
```

```
              precision    recall  f1-score   support

      Setosa       1.00      1.00      1.00        16
  Versicolor       0.00      0.00      0.00        18
   Virginica       0.38      1.00      0.55        11

    accuracy                           0.60        45
   macro avg       0.46      0.67      0.52        45
weighted avg       0.45      0.60      0.49        45

[[16  0  0]
 [ 0  0 18]
 [ 0  0 11]]
```

In [ ]:
```

```