Model used - custom sequential layer embeddings - 200x100 - output(2)

In [12]:
```
dep_var = 'Survived'
cat_var = ['Pclass','Sex','Ticket','Cabin','Embarked','Title']
cont_var = ['Age','Fare','SibSp', 'Parch']
procs = [FillMissing, Categorify, Normalize]
```

In [13]:
```
test = TabularList.from_df(df_test,cat_names=cat_var, cont_names=cont_var, procs=procs,path=path)
```

In [14]:
```
train = (TabularList.from_df(df, cat_names=cat_var, cont_names=cont_var, procs=procs, path=path)
                            .split_by_idx(list(range(0,200)))
                            .label_from_df(cols=dep_var)
                            .add_test(test)
                            .databunch())
```

In [330]:
```
train.show_batch(1)
```

Pclass	Sex	Ticket	Cabin	Embarked	Title	Age	Fare	SibSp	Parch	target
3	female	365226	N	Q	Miss	-0.9165	-0.5077	-0.4610	-0.4658	0

In [15]:
```
learn = tabular_learner(train, layers=[200,100], metrics=accuracy, emb_drop=0.1)
```

In [16]:
```
learn.model
```

Out[16]:
```
TabularModel(
  (embeds): ModuleList(
    (0): Embedding(4, 3)
    (1): Embedding(3, 3)
    (2): Embedding(553, 55)
    (3): Embedding(127, 24)
    (4): Embedding(4, 3)
    (5): Embedding(17, 8)
  )
  (emb_drop): Dropout(p=0.1)
  (bn_cont): BatchNorm1d(4, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (layers): Sequential(
    (0): Linear(in_features=100, out_features=200, bias=True)
    (1): ReLU(inplace)
    (2): BatchNorm1d(200, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Linear(in_features=200, out_features=100, bias=True)
    (4): ReLU(inplace)
    (5): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): Linear(in_features=100, out_features=2, bias=True)
  )
)
```

Basic structures

a. It is a simple 4 layer Neural Network.
    i. Embeddings are created out of categorical inputs and contiguous inputs are taken as such.
    ii. PCA must have been used to create embeddings out of categorical data to identify the minimum numbers of features to represent a column of categorical data.
b. ReLU to introduce non-linearity
c. BatchNormalization to introduce regularization in model.
d. One more thing used for regularization is the dropout.
e. Finally output is binary classification.

Dataset:

a. Titanic kaggle competition dataset
    i. https://www.kaggle.com/c/titanic/data
b. Categories - Survived or not

c. Validation set picked is a range of rows from input. Also tried taking the validation set from random set with percentage 20%.

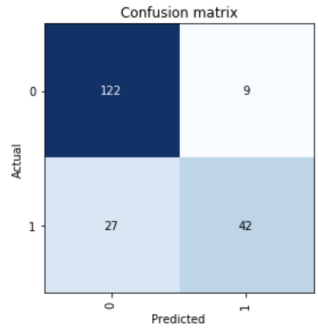
Training steps-

a. Learn.lr_find() give 5e-2 as best learning rate.
b. Used it to fit the learn. Got 80-85% accuracy with 1 epoch.

Confusion matrix

```
In [21]: interp = ClassificationInterpretation.from_learner(learn)

In [22]: interp.plot_confusion_matrix()
```



Observation

a. Panda is used for taking in csv file.
b. Data is divided in contiguous and categorical.
c. Improved the result test accuracy by better cleaning the data.
    i. Filled the nan of contiguous input variables with median.
    ii. Extracted the 'Title' information from 'Name'.
d. More layers are not needed since the training loss was continuously decreasing.
    i. Training loss didn't reach to a minimum value, so it means model is complex enough as data is trying to fit and fit more.
    ii. More data could help to generalize the model further but we don't have that.
e. Tabular data has no show_top_losses function in fastai.

Concepts explored -

a. First actual data from kaggle to test.
b. Used panda more extensively to create different columns and save csv files
c. Used numpy to compare the result values with given values.

Library used - fastai.tabular

Conclusion:

a. Explored a binary classification problem using deep learning technique for tabular data. Used kaggle titanic competition and ranked ~5000 out of 11250. I.e. top 50% with accuracy of 78%

Future work:

a. Explore other machine learning methods for tabular data.
b. What more improvements can be done for better results on tabular data.