Visualizing the trained filters

In [1]:
```
# some startup!
import numpy as np
import matplotlib
# This is needed to save images
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import torch
```

In [2]:
```
# load the model saved by train.py
# This will be an instance of models.convnet.CNN.
# NOTE: You may need to change this file name.
convnet_model = torch.load('convnet.pt')
```

```
/opt/anaconda3/lib/python3.7/site-packages/torch/serialization.py:593: So
urceChangeWarning: source code of class 'models.convnet.CNN' has changed.
you can retrieve the original source code by accessing the object's sourc
e attribute or set `torch.nn.Module.dump_patches = True` and use the patc
h tool to revert the changes.
  warnings.warn(msg, SourceChangeWarning)
```

In [3]:
```
# collect all the weights
w = None
################################################################################
# TODO: Extract the weight matrix (without bias) from convnet_model, conver
# it to a numpy array with shape (10, 32, 32, 3), and assign this array to
# The first dimension should be for channels, then height, width, and color
# This step depends on how you implemented models.convnet.CNN.
################################################################################
w = convnet_model.conv1.cpu().weight.data.numpy()
w = np.transpose(w, (0,2,3,1))
################################################################################
#                                END OF YOUR CODE
################################################################################
# obtain min,max to normalize
w_min, w_max = np.min(w), np.max(w)
# classes
classes = ['plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', '
# init figure
fig = plt.figure(figsize=(6,6))
for i in range(10):
    wimg = 255.0*(w[i].squeeze() - w_min) / (w_max - w_min)
    # subplot is (2,5) as ten filters are to be visualized
    fig.add_subplot(2,5,i+1).imshow(wimg.astype('uint8'))
# save fig!
fig.savefig('convnet_filt.png')
print('figure saved')
```

```
figure saved
```

In [4]:
```
# vis_utils.py has helper code to view multiple filters in single image. Us
# neural network.
# import vis_utils
from vis_utils import visualize_grid
# saving the weights is now as simple as:
plt.imsave('convnet_gridfilt.png',visualize_grid(w, padding=3).astype('uint
# padding is the space between images. Make sure that w is of shape: (N,H,W
print('figure saved as a grid!')
```

```
figure saved as a grid!
```

In [ ]:
```

```