In this lab a Geiger counter to study the statistics of radioactive decay.

In [2]:
```
#imports numpy and matplotlib
%pylab inline

#imports gaussian function
from scipy.stats import norm

#imports poisson function
from scipy.stats import poisson
```

```
Populating the interactive namespace from numpy and matplotlib
```

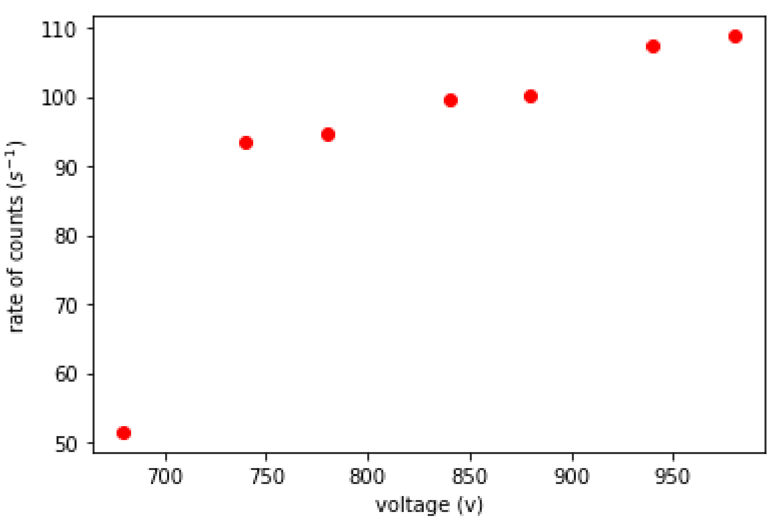
In [3]:
```
#voltage applied to the ssample in volts
voltage = np.array([680, 740, 780, 840, 880, 940, 980])

#counts made over 10 second intevals of testing the sample
count = np.array([515, 935, 948, 996, 1002, 1073, 1088])

#number of counts per second over the the intervals collected
rate = count/10

#plotting the rate vs voltage for te counts from the radioactive sample.
plt.plot(voltage, rate, "ro",label="rate vs. V")
plt.xlabel("voltage (v)")
plt.ylabel("rate of counts ($s^{-1}$)")
plt.show()

#graph description
print('''
Graph of the rate rate of counts from the Geiger counter as a function of the ap
''')
```




```
Graph of the rate rate of counts from the Geiger counter as a function of the ap
plied voltage.
```

In [4]:
```
#120 measurements of counts from Geigar counter for the radioactive source using
foreground = np.array([1015, 1039, 973, 987, 1027, 1051, 1063, 1040, 1103, 1011,
                       939, 981, 1037, 990, 1043, 1055, 1019, 1053, 1016, 1000, 9
                       997, 939, 1031, 1015, 1029, 909, 1038, 1069, 1031, 998, 10
```

```
                      1004, 1061, 986, 1037, 1007, 971, 1038, 1017, 1056, 982, 9
                      995, 1152, 1040, 1006, 1045, 1039, 1016, 1043, 1038, 1002,
                      952, 1079, 1019, 1086, 1054, 1034, 983, 1042, 1037, 927, 9
                      1012, 1044, 1027, 1054, 960, 1048, 1022, 1046, 1004, 1041,
                      1078, 999, 995, 976, 1023, 975, 992, 1078, 994, 1070, 1048
                      1024, 1032, 1003, 997, 1003, 1033, 1000, 1018, 1028, 1039,
                      1057, 978, 1003, 976, 999, 1031, 1008, 967, 1071, 1009, 10



#120 measurements of counts from Geigar counter for the background radiation
#after the radioactive was removed.
background = np.array([5, 1, 3, 5, 9, 6, 3, 1, 3, 4, 2, 6,
                      2, 5, 3, 5, 5, 6, 2, 5, 5, 6, 6, 1,
                      4, 3, 6, 4, 4, 2, 5, 3, 3, 3, 6, 3,
                      10, 1, 8, 5, 5, 6, 7, 3, 2, 5, 2, 6,
                      3, 5, 5, 4, 2, 3, 5, 4, 2, 4, 8, 7,
                      4, 3, 0, 2, 4, 3, 3, 0, 2, 4, 4, 1,
                      3, 7, 1, 3, 5, 3, 7, 2, 3, 4, 2, 8,
                      1, 2, 4, 4, 3, 2, 5, 3, 4, 4, 3, 3,
                      2, 4, 8, 5, 1, 4, 5, 3, 1, 4, 9, 6,
                      7, 5, 4, 2, 2, 5, 4, 3, 6, 6, 4, 7])
```

In [5]:

```
#average number of counts from radioactive sample
fbar = np.average(foreground)
#variance of counts from radioactive sample
fvar = np.var(foreground)
#standard deviation of counts from the radioactive sample
fsig = np.std(foreground)

#average number of counts from background radiation
bbar = np.average(background)
#variance of counts from background radiation
bvar = np.var(background)
#standard deviation of counts from background radiation
bsig = np.std(background)

#statistics for the counts from the sample, and the background information


print("sample count average: ",fbar)
print("sample count variance: ",fvar)
print("sample count standard deviation: ",fsig)
print(" ")
print("background count average: ",bbar)
print("background count variance: ",bvar)
print("background count standard deviation: ",bsig)
print(" ")


#print statements used to assist in creating the histograms
print('sample histogram should range from about ', np.amin(foreground))
print('to about ', np.amax(foreground))
print('and each bin should have size about ',(np.amax(foreground)-np.amin(foregr
print(" ")
print('background histogram should range from about ', np.amin(background))
print('to about ', np.amax(background))
print('and each bin should have size about ',(np.amax(background)-np.amin(backgr
```

```
sample count average:  1018.1416666666667
sample count variance:  1407.7215972222223
sample count standard deviation:  37.519616165710204

background count average:  4.0
background count variance:  3.9833333333333334
background count standard deviation:  1.9958289839896939

sample histogram should range from about  909
to about  1152
and each bin should have size about  12.15

background histogram should range from about  0
to about  10
and each bin should have size about  0.5
```

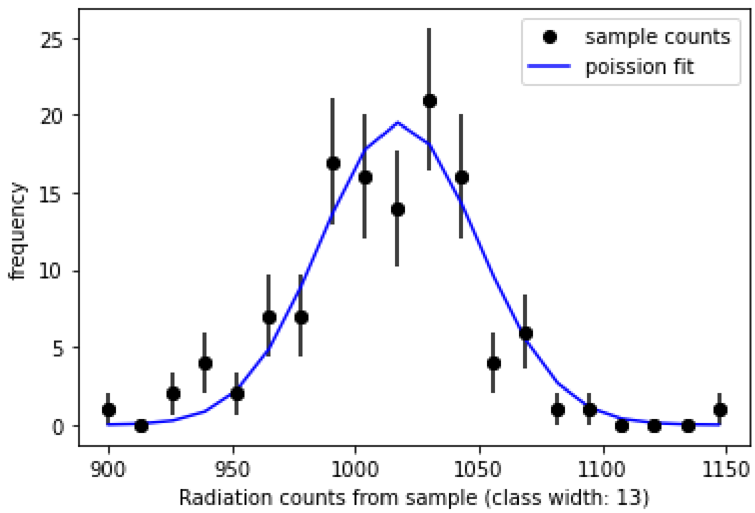
In [6]:

```python
#creating and plotting histogram for the sample data
f_occurrence, f_width= np.histogram(foreground, bins=20,range=(900,1160))
fbin = (1160 - 900)/20
plt.plot(f_width[:-1],f_occurrence, "ko",label="sample counts")
f_fluxuation = f_occurrence**0.5
plt.errorbar(f_width[:-1], f_occurrence, f_fluxuation, fmt="ko")

#creating and plotting poisson distribution fit to sample data
fx = f_width[:-1]
ffish = fbin*np.size(foreground)*poisson.pmf(fx, fbar)
plt.plot(fx,ffish,"b-",label="poission fit")

#adding necesary information to the graph
plt.xlabel("Radiation counts from sample (class width: 13)")
plt.ylabel("frequency")
plt.legend()
plt.show()

#graph description
print('''
poisson distribution fit to the 120 counts from the
Geiger counter, using 900v, for the radioactive sample.
''')
```




```
poisson distribution fit to the 120 counts from the
Geiger counter, using 900v, for the radioactive sample.
```

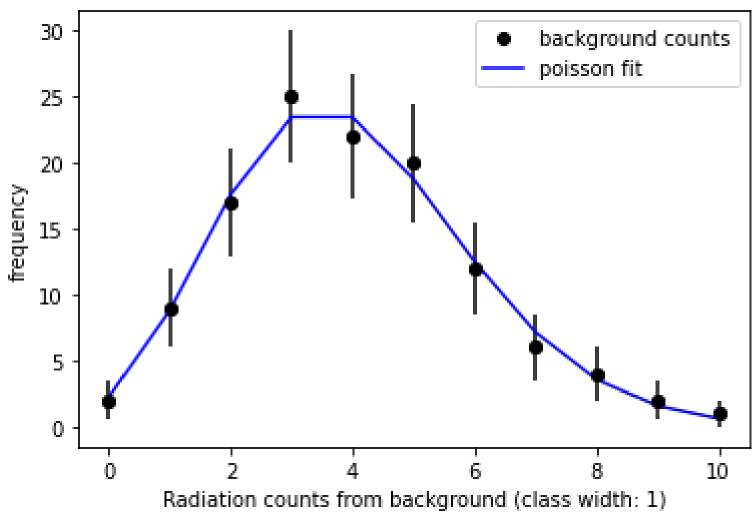
In [7]:

```
#creating and plotting histogram for the background data
b_occurrence, b_width = np.histogram(background, bins=11, range=(0,11))
bbin = (11 - 0)/11
plt.plot(b_width[:-1], b_occurrence, "ko",label="background counts")
b_fluxuation = b_occurrence**0.5
plt.errorbar(b_width[:-1], b_occurrence, b_fluxuation, fmt="ko")

#creating and plotting poisson distribution fit to sample data
bx = b_width[:-1]
bfish = bbin*np.size(background)*poisson.pmf(bx, bbar)
plt.plot(bx,bfish,"b-",label="poisson fit")

#adding necesary information to the graph
plt.xlabel("Radiation counts from background (class width: 1)")
plt.ylabel("frequency")
plt.legend()
plt.show()

#graph description
print('''
poisson distribution fit to the 120 counts from the
Geiger counter, using 900v, for the background radiation.
''')
```




```
poisson distribution fit to the 120 counts from the
Geiger counter, using 900v, for the background radiation.
```

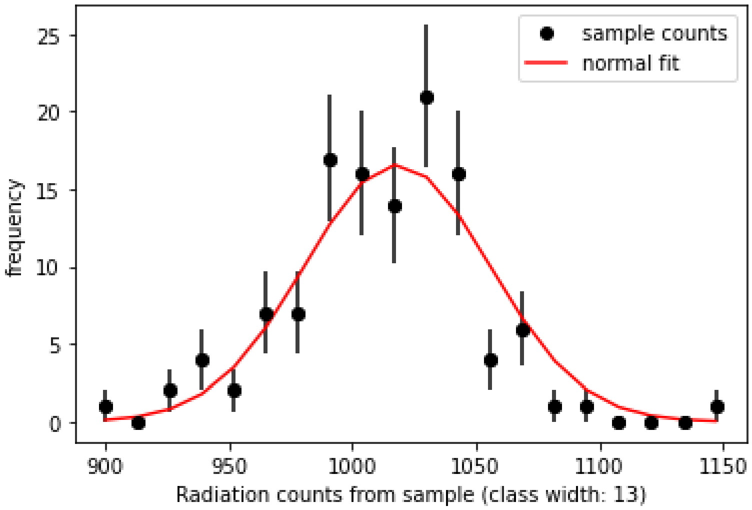
In [8]:

```
#plotting sample data as a histogram
plt.plot(f_width[:-1],f_occurrence, "ko",label="sample counts")
plt.errorbar(f_width[:-1], f_occurrence, f_fluxuation, fmt="ko")

#creating and plotting normal distribution fit to sample data
fnorm = fbin*np.size(foreground)*norm.pdf(fx, loc=fbar, scale=fsig)
plt.plot(fx,fnorm,"r-",label="normal fit")

#adding necesary information to the graph
plt.xlabel("Radiation counts from sample (class width: 13)")
plt.ylabel("frequency")
plt.legend()
plt.show()
```

```
#graph description
print('''
normal distribution fit to the 120 counts from the
Geiger counter, using 900v, for the radioactive sample.
''')
```




```
normal distribution fit to the 120 counts from the
Geiger counter, using 900v, for the radioactive sample.
```

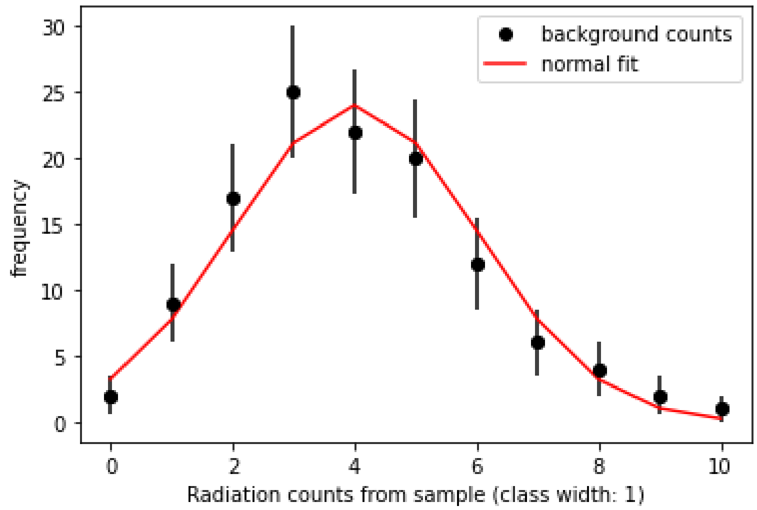
In [9]:

```
#plotting backgroud data as a histogram
plt.plot(b_width[:-1], b_occurrence, "ko",label="background counts")
plt.errorbar(b_width[:-1], b_occurrence, b_fluxuation, fmt="ko")

#creating and plotting normal distribution fit to background data
bnorm = np.size(background)*norm.pdf(bx, loc=bbar, scale=bsig)
plt.plot(bx,bnorm,"r-",label="normal fit")

#adding necesary information to the graph
plt.xlabel("Radiation counts from sample (class width: 1)")
plt.ylabel("frequency")
plt.legend()
plt.show()

#graph description
print('''
normal distribution fit to the 120 counts from the
Geiger counter, using 900v, for the background radiation.
''')
```




```
normal distribution fit to the 120 counts from the
Geiger counter, using 900v, for the background radiation.
```

In [ ]: