基本用法

本章知识点归纳如下：

1.散点图：plt.scatter()

2.柱状图：plt.bar()

3.等高线图：plt.contourf()

4.在等高线图中增加label：plt.clabel()

5.矩阵画图：plt.imshow()

6.在随机矩阵图中增加colorbar：plt.colorbar()

散点图

首先，先引入matplotlib.pyplot简写作plt,再引入模块numpy用来产生一些随机数据。

1.数据生成

生成1024个呈标准正态分布的二维数据组 (平均数是0，方差为1) 作为一个数据集，并图像化这个数据集。每一个点的颜色值用T来表示：

In [1]:

```
import matplotlib.pyplot as plt
import numpy as np

n = 1024    # data size
X = np.random.normal(0, 1, n) # 每一个点的X值
Y = np.random.normal(0, 1, n) # 每一个点的Y值
T = np.arctan2(Y,X) # for color value
```

2.画图：

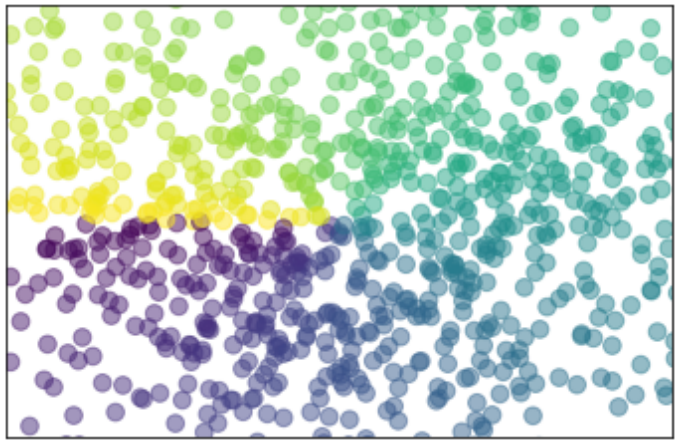
数据集生成完毕，现在来用 plt.scatter 画出这个点集，输入X和Y作为location，size=75，颜色为T，color map用默认值，透明度alpha 为 50%。 x轴显示范围定位(-1.5，1.5)，并向xtick()函数传入空集()来隐藏x坐标轴，y轴同理：

In [2]:

```
plt.scatter(X, Y, s=75, c=T, alpha=.5)

plt.xlim(-1.5, 1.5)
plt.xticks(())  # ignore xticks
plt.ylim(-1.5, 1.5)
plt.yticks(())  # ignore yticks

plt.show()
```



柱状图

柱状图是在数据分析过程中最为常用的图表，折线图和饼图能够表达的信息，柱状图都能够表达。在学术报告或工作场景下，大家应尽量使用柱状图来代替折线图与饼图。下面，我们就开始吧~

1.数据生成：

首先生成画图数据，向上向下分别生成2组数据，X为0到11的整数 ， Y是相应的均匀分布的随机数据。

2.画图：

使用的函数是plt.bar，参数为X和Y，X代表横坐标，即柱形的位置，Y代表纵坐标，即柱形的高度。

In [16]:

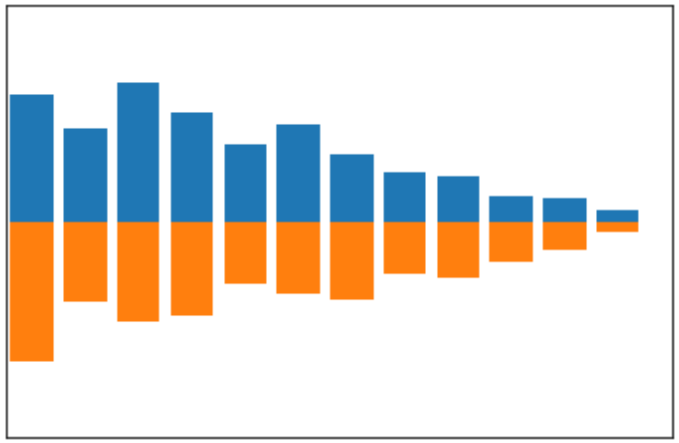
```
import matplotlib.pyplot as plt
import numpy as np

n = 12
X = np.arange(n)
Y1 = (1 - X / float(n)) * np.random.uniform(0.5, 1.0, n)
Y2 = (1 - X / float(n)) * np.random.uniform(0.5, 1.0, n)

plt.bar(X, +Y1)
plt.bar(X, -Y2)

plt.xlim(-.5, n)
plt.xticks(())
plt.ylim(-1.25, 1.25)
plt.yticks(())

plt.show()
```



3.修改颜色和数据标签

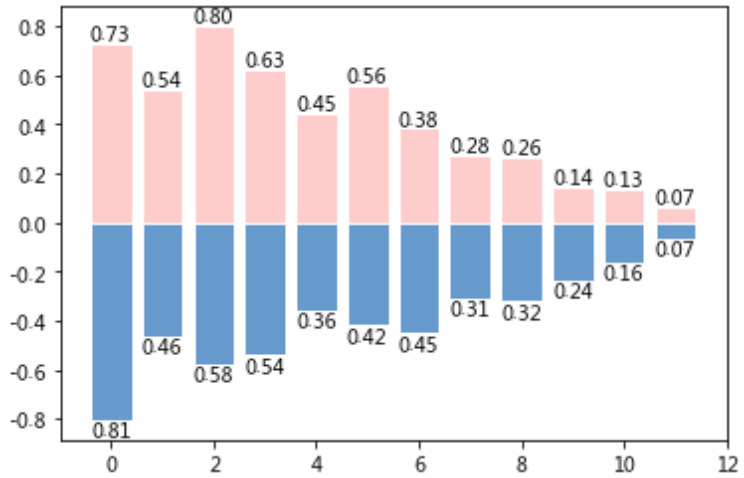
如果小伙伴们想要改变柱状图的颜色，并且希望每个柱形上方能够显示该项数值该怎么做呢？我们可以用 plt.bar 函数中的facecolor参数设置柱状图主体颜色，用edgecolor参数设置边框颜色；而函数 plt.text 可以帮助我们在柱体上方（下方）加上数值：用%.2f保留两位小数，用ha='center'设置横向居中对齐，用va='bottom'设置纵向底部（顶部）对齐。

In [18]:

```
plt.bar(X, +Y1, facecolor='#FFCCCC', edgecolor='white')
plt.bar(X, -Y2, facecolor='#6699CC', edgecolor='white')

for x, y in zip(X, Y1):
    # ha: horizontal alignment
    # va: vertical alignment
    plt.text(x, y , '%.2f' % y, ha='center', va='bottom')

for x, y in zip(X, Y2):
    # ha: horizontal alignment
    # va: vertical alignment
    plt.text(x, -y , '%.2f' % y, ha='center', va='top')
```



等高线图

1.数据生成

数据集即三维点 (x,y) 和对应的高度值，共有256个点。高度值使用一个 height function f(x,y) 生成。 x, y 分别是在区间 [-3,3] 中均匀分布的256个值，并用meshgrid在二维平面中将每一个x和每一个y分别对应起来，编织成栅格:

In [7]:

```
def f(x,y):
    # the height function
    return (1 - x / 2 + x**5 + y**3) * np.exp(-x**2 -y**2)

n = 256
x = np.linspace(-3, 3, n)
y = np.linspace(-3, 3, n)
X,Y = np.meshgrid(x, y)
```

2.画图:

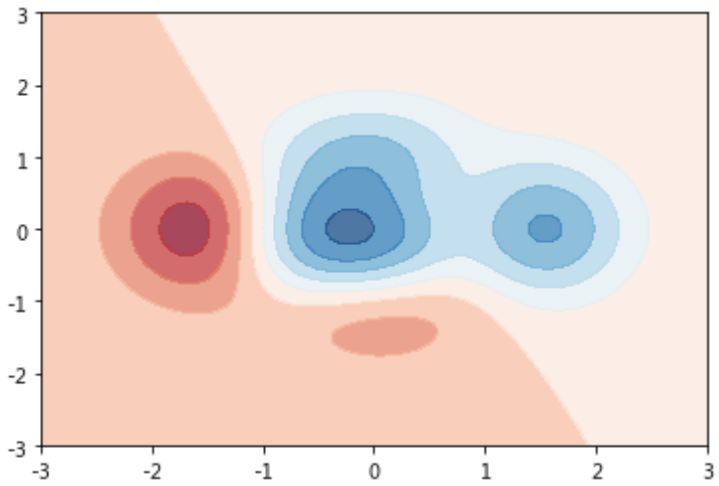
接下来进行颜色填充。使用函数plt.contourf把颜色加进去，位置参数分别为：X, Y, f(X,Y)。8代表等高线的密集程度，这里被分为10个部分。如果是0，则图像被一分为二。透明度为0.75，并将 f(X,Y) 的值对应到color map的RdBu组中寻找对应颜色。大家可能并不能直观理解 colormap ，它可以将颜色和数字进行映射。如果暂时不能理解的话也没有关系，我们可以将其想象成matplotlib为我们提供的配色方案，大家可以查看此链接选择自己喜欢的配色方案应用在自己的图上。

In [8]:

```
# use plt.contourf to filling contours
# X, Y and value for (X,Y) point
plt.contourf(X, Y, f(X, Y), 8, alpha=.75, cmap=plt.cm.RdBu)
```

Out[8]:

```
<matplotlib.contour.QuadContourSet at 0x7f79c1734b38>
```

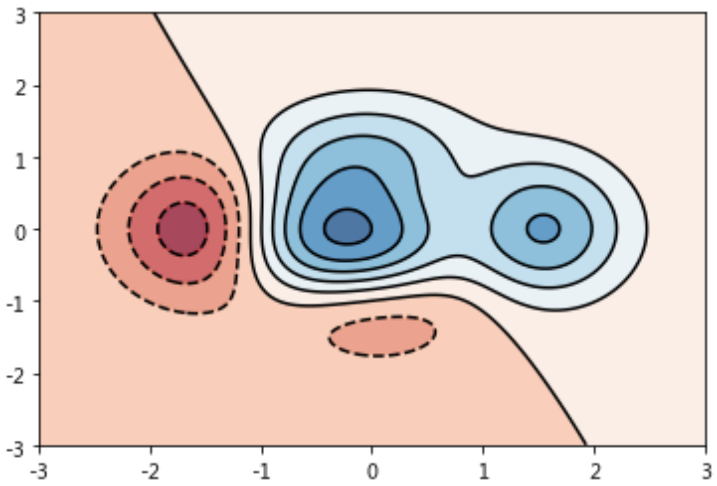


接下来进行等高线绘制。使用plt.contour函数划线。位置参数为：X, Y, f(X,Y)。颜色选黑色，线条宽度选0.5。现在的结果如下图所示，只有颜色和线条，还没有数值Label：

In [9]:

```
# use plt.contour to add contour lines
plt.contourf(X, Y, f(X, Y), 8, alpha=.75, cmap=plt.cm.RdBu)
C = plt.contour(X, Y, f(X, Y), 8, colors='black', linewidth=.5)
```

```
/opt/conda/lib/python3.5/site-packages/matplotlib/contour.py:960: UserWarning: The following kwargs were not used by contour: 'linewidth'
  s)
```



3.添加高度数字：

最后我们要通过 plt.clabel() 在等高线上加入高度数值，即加入Label，其中参数 inline 控制是否将 Label 画在线里面，fontsize 设置字体大小为10。并将坐标轴隐藏：

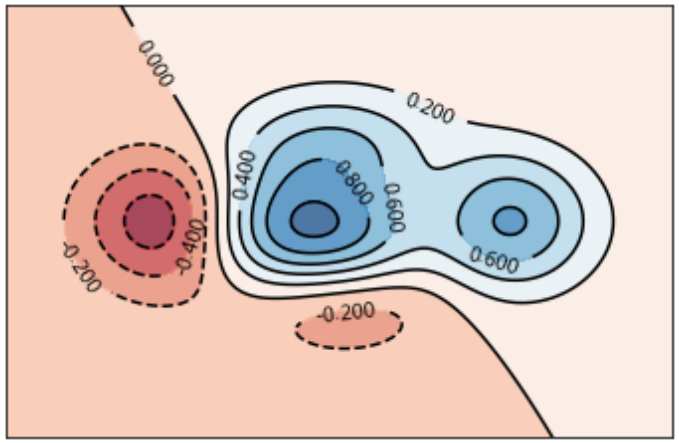
In [10]:

```
plt.contourf(X, Y, f(X, Y), 8, alpha=.75, cmap=plt.cm.RdBu)
C = plt.contour(X, Y, f(X, Y), 8, colors='black', linewidth=.5)
plt.clabel(C, inline=True, fontsize=10)
plt.xticks(())
plt.yticks(())
```

```
/opt/conda/lib/python3.5/site-packages/matplotlib/contour.py:960: UserWarning: The following kwargs were not used by contour: 'linewidth'
  s)
```

Out[10]:

```
([], <a list of 0 Text yticklabel objects>)
```



随机矩阵画图

这一节我们讲解怎样在matplotlib中打印出图像。这里我们打印出的是纯粹的数字，而非自然图像。

1.数据生成

首先生成一个 3x3 的 2D-array ， 也就是三行三列的格子，array 中的每个值经过colormap与一个颜色对应并填充在格子中

In [19]:

```
a = np.array([0.313660827978, 0.365348418405, 0.423733120134,
              0.365348418405, 0.439599930621, 0.525083754405,
              0.423733120134, 0.525083754405, 0.651536351379]).reshape(3,3)
```

2.画图：

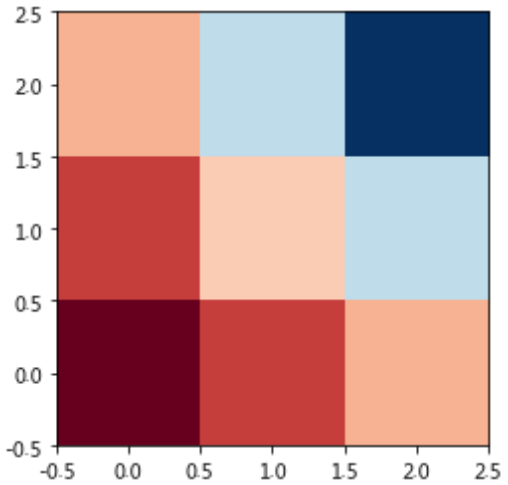
我们之前选cmap的参数时用的是：cmap=plt.cmap.RdBu，而现在，我们可以直接用单引号传入参数。 origin='lower'代表的就是选择的原点的位置。而 interpolation 表示画图方式，从该链接可看到matplotlib官网上对于内插法的不同方法的描述。这里我们使用的是内插法中的 Nearest-neighbor 的方法，其他的方式也都可以随意取选。

In [20]:

```
plt.imshow(a, interpolation='nearest', cmap='RdBu', origin='lower')
```

Out[20]:

```
<matplotlib.image.AxesImage at 0x7f79c1641b38>
```



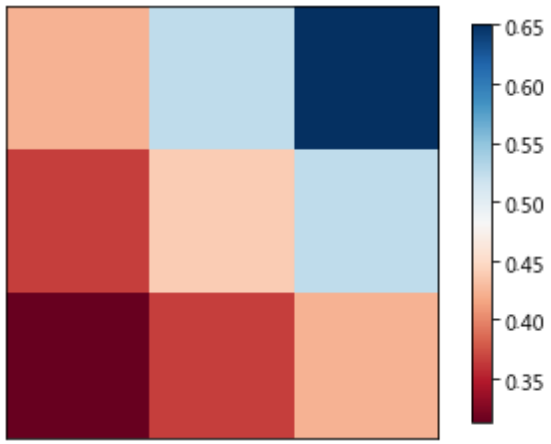
3.增加colorbar

下面我们添加一个colorbar，它可以为我们显示不同颜色的区块所对应的具体数值。其中shrink参数可以用来调整 colorbar 的长度，这里我们使colorbar的长度变短为原来的92%，这样我们2D图像就创建完毕了：

In [21]:

```
plt.imshow(a, interpolation='nearest', cmap='RdBu', origin='lower')
plt.colorbar(shrink=.92)

plt.xticks(())
plt.yticks(())
plt.show()
```



练一练

现在，小伙伴们可以尝试用上述方法对豆瓣电影数据集进行分析。请根据数据集中的表格'电影影评.csv'画出电影星级分布图。横坐标为电影的评分星级，分别为1，2，3，4，5；纵坐标为该星级下的电影数量。 提示：使用value_counts()函数对不同星级的电影数量进行计算

In [22]:

```
#答案:
import pandas as pd
data = pd.read_csv('/home/kesci/input/movie_douban/电影影评.csv')
data = data[data['星级']<=5]
data_distri = data['星级'].value_counts()
```

```
/opt/conda/lib/python3.5/site-packages/IPython/core/interactiveshell.py:2785: DtypeWarning: Columns (12,13,14,15,16,17,18,19,20,21,22,23,24,25,28,33
  interactivity=interactivity, compiler=compiler, result=result)
```

In [23]:

```
plt.figure(figsize=(6,4))
plt.bar(data_distri.index, data_distri.values)
plt.ylim(0, 60000)
#设置数值标签
x = np.array(list(data_distri.index))
y = np.array(list(data_distri.values))
for a,b in zip(x,y):
    plt.text(a, b+500, '%.0f' % b, ha='center', va= 'bottom',fontsize=10)
```