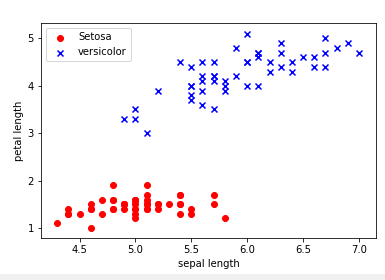
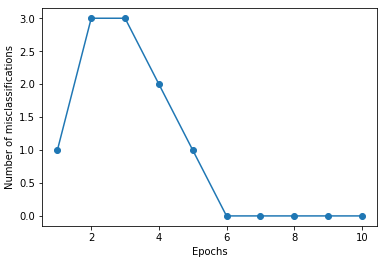
**基于鸢尾花数据集训练感知器模型**

1. 为了方便实现感知器算法，从鸢尾花数据集中挑选了山鸢尾和变色鸢尾两种花的信息作为测试数据。虽然感知器并不将数据样本的特征限制为2个，为了可视化方便，我们只考虑萼片长度和花瓣长度这两个特征。
2. 使用pd.read\_csv()方法读取信息iris信息。因为我们只需要第一列和第三列的数据作为特征集X，而存储对应类标的第五列作为类标向量y。为了更好的实现二分类算法，我们将类标标准化，将山鸢尾的类标设为-1，变色鸢尾设为1。并用散点图对这些数据进行可视化。

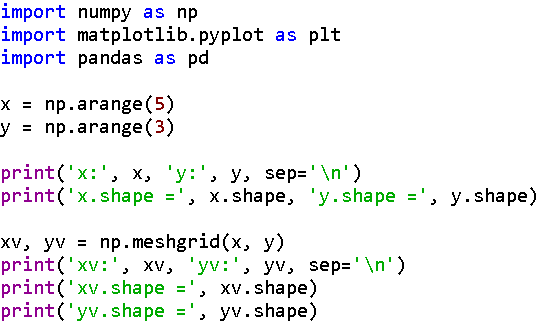


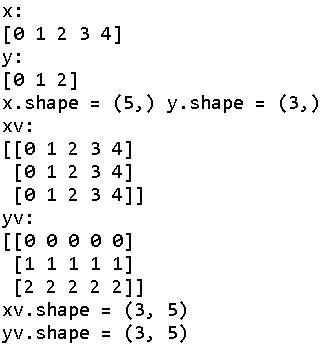
1. 为了测试感知器的分类效果，绘制每次迭代的错误分类数量折线图：



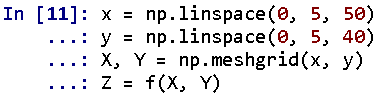
1. 等高线图可以用plt.contour()函数来创建，它需要三个参数：x轴、y轴、z轴三个坐标轴的网格数据。x轴和y轴表示图形中的位置，而z轴将通过等高线的等级来表示，即z表示高度，在这里用一个函数表示。用np.meshgrid( )函数来准备这些数据是最简单的方法，它可以从一维数组构建二维网格数据.

np.meshgrid(x,y)将向量x和y定义的区域，转换成矩阵X和Y，其中矩阵X的行向量是x的简单复制，而Y的列向量是向量y的简单复制。（假设x是长度为m的向量，y是长度为n向量，最终生成的矩阵X和Y都是n行m列）。meshgrid()在二维平面中将每个x和每个y分别对应起来，编织成栅格。



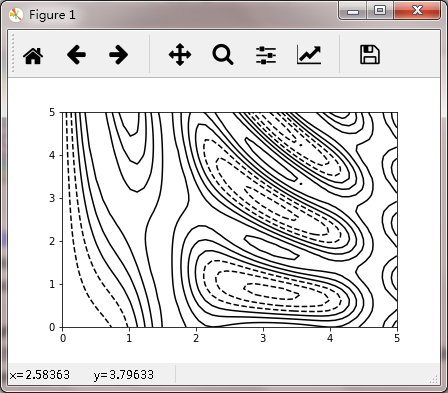


这里X=xv,Y=yv。尝试构建一个：

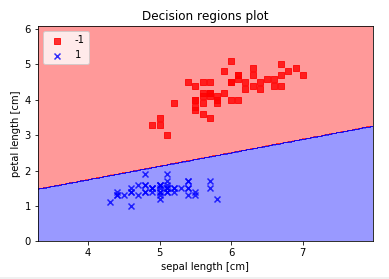


现在来绘制标准的线形等高线图：





1. 通过一个简单的函数plot\_decision\_regions()来实现对二维数据集决策边界的可视化。运行代码后的得到的图像如下：



注意：这里将山鸢尾坐标标准化为1，变色鸢尾坐标标准化为-1.

1. 虽然感知器模型可以完美的划分两个类别的花，但是感知器所面临的最大的问题是算法的收敛。从数学角度可以证明：如果两个类别可以通过线性超平面进行划分，则感知器算法一定收敛。但是，如果两个类别无法通过线性判定边界完全正确地划分，则权重会不断更新。所以为了预防此类事件，通常事先设置权重更新的最大迭代次数，即Perception类中的n\_iter。