逻辑斯蒂回归

【关键词】Logistics函数，最大似然估计，梯度下降法

1、Logistics回归的原理

利用Logistics回归进行分类的主要思想是：根据现有数据对分类边界线建立回归公式，以此进行分类。这里的“回归” 一词源于最佳拟合，表示要找到最佳拟合参数集。

训练分类器时的做法就是寻找最佳拟合参数，使用的是最优化算法。接下来介绍这个二值型输出分类器的数学原理

Logistic Regression和Linear Regression的原理是相似的，可以简单的描述为这样的过程：

（1）找一个合适的预测函数，一般表示为h函数，该函数就是我们需要找的分类函数，它用来预测输入数据的判断结果。这个过程是非常关键的，需要对数据有一定的了解或分析，知道或者猜测预测函数的“大概”形式，比如是线性函数还是非线性函数。

（2）构造一个Cost函数（损失函数），该函数表示预测的输出（h）与训练数据类别（y）之间的偏差，可以是二者之间的差（h-y）或者是其他的形式。综合考虑所有训练数据的“损失”，将Cost求和或者求平均，记为J(θ)函数，表示所有训练数据预测值与实际类别的偏差。

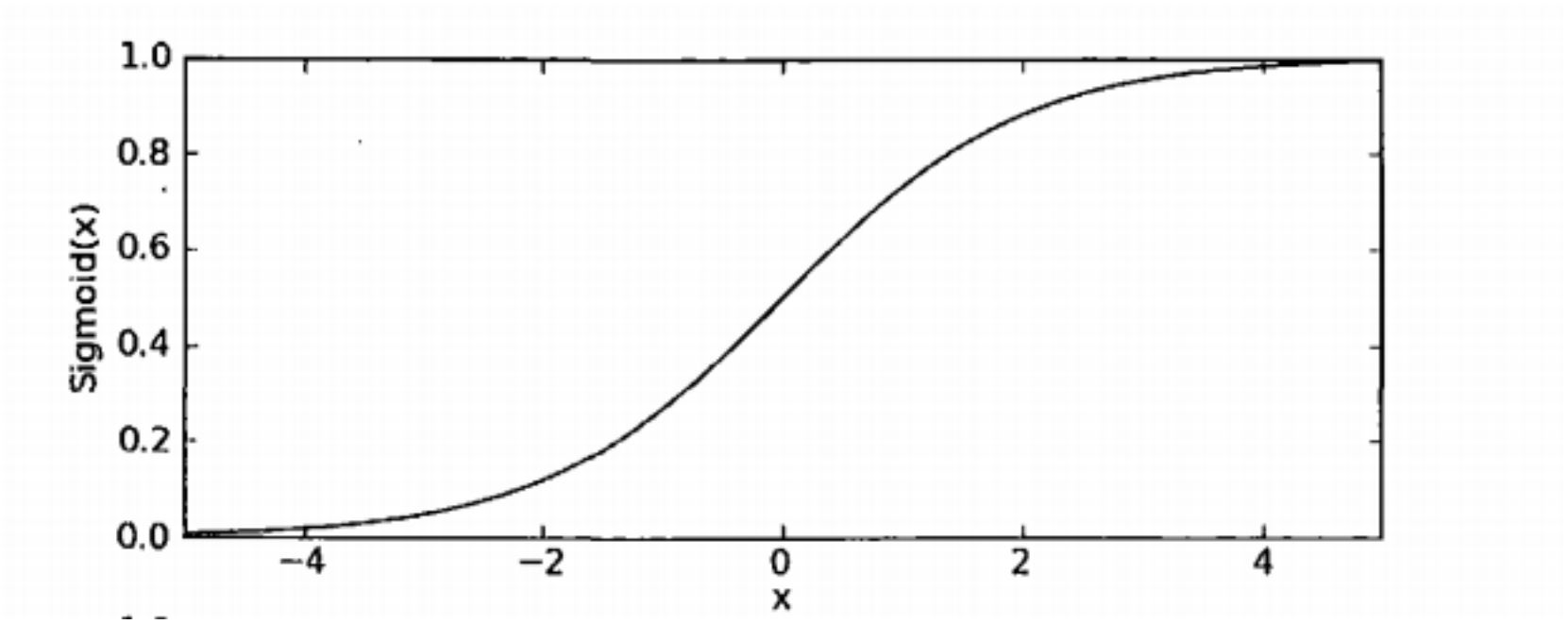
（3）显然，J(θ)函数的值越小表示预测函数越准确（即h函数越准确），所以这一步需要做的是找到J(θ)函数的最小值。找函数的最小值有不同的方法，Logistic Regression实现时有梯度下降法（Gradient Descent）。

1) 构造预测函数

Logistic Regression虽然名字里带“回归”，但是它实际上是一种分类方法，用于两分类问题（即输出只有两种）。首先需要先找到一个预测函数（h），显然，该函数的输出必须是两类值（分别代表两个类别），所以利用了*Logistic函数（或称为Sigmoid函数）*，函数形式为：

$$\sigma(z)=\frac{1}{1+e^{-z}}$$

该函数形状为：



预测函数可以写为：

$$h_\theta(x)=g(\theta^T x)=\frac{1}{1+e^{-\theta^T x}} \quad (3)$$

2）构造损失函数

Cost函数和J(θ)函数是基于*最大似然估计*推导得到的。

每个样本属于其真实标记的概率，即似然函数，可以写成：

$$P(y \mid x;\theta) = (h_\theta(x))^{y}(1-h_\theta(x))^{1-y} \tag{7}$$

所有样本都属于其真实标记的概率为

$$L(\theta) = \prod_{i=1}^{m} P(y^{(i)} \mid x^{(i)};\theta)$$

$$= \prod_{i=1}^{m} (h_\theta(x^{(i)}))^{y^{(i)}} (1-h_\theta(x^{(i)}))^{1-y^{(i)}} \tag{8}$$

对数似然函数为

$$l(\theta) = \log L(\theta)$$

$$= \sum_{i=1}^{m} \left(y^{(i)} \log h_\theta(x^{(i)}) + \left(1-y^{(i)}\right) \log(1-h_\theta(x^{(i)})) \right) \tag{9}$$

最大似然估计就是要求得使l(θ)取最大值时的θ，其实这里可以使用梯度上升法求解，求得的θ就是要求的最佳参数

3) 梯度下降法求J(θ)的最小值

求J(θ)的最小值可以使用*梯度下降法*，根据梯度下降法可得θ的更新过程:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta), \quad (j = 0 \ldots n) \tag{11}$$

式中为α学习步长，下面来求偏导：

$$\frac{\partial}{\partial\theta_j}J(\theta)=-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\frac{1}{h_\theta(\mathrm{x}^{(\mathrm{i})})}\frac{\partial}{\partial\theta_j}h_\theta(\mathrm{x}^{(\mathrm{i})})-(1-y^{(i)})\frac{1}{1-h_\theta(\mathrm{x}^{(\mathrm{i})})}\frac{\partial}{\partial\theta_j}h_\theta(\mathrm{x}^{(\mathrm{i})})\right)$$

$$=-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\frac{1}{g(\theta^T x^{(\mathrm{i})})}-(1-y^{(i)})\frac{1}{1-g(\theta^T x^{(\mathrm{i})})}\right)\frac{\partial}{\partial\theta_j}g(\theta^T x^{(\mathrm{i})})$$

$$=-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\frac{1}{g(\theta^T x^{(\mathrm{i})})}-(1-y^{(i)})\frac{1}{1-g(\theta^T x^{(\mathrm{i})})}\right)g(\theta^T x^{(\mathrm{i})})\left(1-g(\theta^T x^{(\mathrm{i})})\right)\frac{\partial}{\partial\theta_j}\theta^T x^{(\mathrm{i})}$$

$$=-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\left(1-g(\theta^T x^{(\mathrm{i})})\right)-(1-y^{(i)})g(\theta^T x^{(\mathrm{i})})\right)x_j^{(\mathrm{i})}$$

$$=-\frac{1}{m}\sum_{i=1}^{m}\left(y^{i}-g(\theta^T x^{(\mathrm{i})})\right)x_j^{(\mathrm{i})}$$

$$=-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}-h_\theta(\mathrm{x}^{(\mathrm{i})})\right)x_j^{(\mathrm{i})}$$

$$=\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(\mathrm{x}^{(\mathrm{i})})-y^{(i)}\right)x_j^{(\mathrm{i})}$$

（12）

上式求解过程中用到如下的公式：

$$f(x)=\frac{1}{1+e^{g(\mathrm{x})}}$$

$$\frac{\partial}{\partial x}f(x)=\frac{1}{\left(1+e^{g(\mathrm{x})}\right)^2}e^{g(\mathrm{x})}\frac{\partial}{\partial x}g(x)$$

（13）

$$=\frac{1}{1+e^{g(\mathrm{x})}}\frac{e^{g(\mathrm{x})}}{1+e^{g(\mathrm{x})}}\frac{\partial}{\partial x}g(x)$$

$$=f(x)\left(1-f(x)\right)\frac{\partial}{\partial x}g(x)$$

因此，θ的更新过程可以写成:

$$\theta_j:=\theta_j-\alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(\mathrm{x}^{(\mathrm{i})})-y^{(i)}\right)x_j^{(\mathrm{i})},\quad(j=0\ldots n)\qquad(14)$$

因为式中α本来为一常量，所以1/m一般将省略，所以最终的θ更新过程为：

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} \left(h_\theta(\mathrm{x}^{(i)}) - y^{(i)} \right) x_j^{(i)}, \quad (j = 0 \ldots n) \qquad (15)$$

Type *Markdown* and LaTeX: $\alpha^2 \alpha^2$

2、实战

sklearn.linear_model.LogisticRegression(penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='liblinear', max_iter=100, multi_class='ovr', verbose=0, warm_start=False, n_jobs=1)

solver参数的选择：

- “liblinear”：小数量级的数据集
- “lbfgs”, “sag” or “newton-cg”：大数量级的数据集以及多分类问题
- “sag”：极大的数据集

In [1]:

```
# 逻辑斯提 Logistic
# 是一个线性回归模型，处理二分类问题
# 概率论
# 对分类边界建立回归公式
# 不能处理回归问题
```

1) 手写数字数据集的分类

使用KNN与Logistic回归两种方法

In [2]:

```
from sklearn.datasets import load_digits
```

In [3]:

```
digits = load_digits()
```

In [4]:

```
digits
```

Out[4]:

```
{'data': array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
        [ 0.,  0.,  0., ..., 10.,  0.,  0.],
        [ 0.,  0.,  0., ..., 16.,  9.,  0.],
```

```
        ...,
        [ 0.,  0.,  1., ...,  6.,  0.,  0.],
        [ 0.,  0.,  2., ..., 12.,  0.,  0.],
        [ 0.,  0., 10., ..., 12.,  1.,  0.]]),
'target': array([0, 1, 2, ..., 8, 9, 8]),
'target_names': array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
'images': array([[[ 0.,  0.,  5., ...,  1.,  0.,  0.],
        [ 0.,  0., 13., ..., 15.,  5.,  0.],
        [ 0.,  3., 15., ..., 11.,  8.,  0.],
        ...,
        [ 0.,  4., 11., ..., 12.,  7.,  0.],
        [ 0.,  2., 14., ..., 12.,  0.,  0.],
        [ 0.,  0.,  6., ...,  0.,  0.,  0.]],

       [[ 0.,  0.,  0., ...,  5.,  0.,  0.],
        [ 0.,  0.,  0., ...,  9.,  0.,  0.],
        [ 0.,  0.,  3., ...,  6.,  0.,  0.],
        ...,
        [ 0.,  0.,  1., ...,  6.,  0.,  0.],
        [ 0.,  0.,  1., ...,  6.,  0.,  0.],
        [ 0.,  0.,  0., ..., 10.,  0.,  0.]],

       [[ 0.,  0.,  0., ..., 12.,  0.,  0.],
        [ 0.,  0.,  3., ..., 14.,  0.,  0.],
        [ 0.,  0.,  8., ..., 16.,  0.,  0.],
        ...,
        [ 0.,  9., 16., ...,  0.,  0.,  0.],
        [ 0.,  3., 13., ..., 11.,  5.,  0.],
        [ 0.,  0.,  0., ..., 16.,  9.,  0.]],

       ...,

       [[ 0.,  0.,  1., ...,  1.,  0.,  0.],
        [ 0.,  0., 13., ...,  2.,  1.,  0.],
        [ 0.,  0., 16., ..., 16.,  5.,  0.],
        ...,
        [ 0.,  0., 16., ..., 15.,  0.,  0.],
        [ 0.,  0., 15., ..., 16.,  0.,  0.],
        [ 0.,  0.,  2., ...,  6.,  0.,  0.]],

       [[ 0.,  0.,  2., ...,  0.,  0.,  0.],
        [ 0.,  0., 14., ..., 15.,  1.,  0.],
        [ 0.,  4., 16., ..., 16.,  7.,  0.],
        ...,
        [ 0.,  0.,  0., ..., 16.,  2.,  0.],
        [ 0.,  0.,  4., ..., 16.,  2.,  0.],
        [ 0.,  0.,  5., ..., 12.,  0.,  0.]],

       [[ 0.,  0., 10., ...,  1.,  0.,  0.],
        [ 0.,  2., 16., ...,  1.,  0.,  0.],
        [ 0.,  0., 15., ..., 15.,  0.,  0.],
        ...,
        [ 0.,  4., 16., ..., 16.,  6.,  0.],
        [ 0.,  8., 16., ..., 16.,  8.,  0.],
        [ 0.,  1.,  8., ..., 12.,  1.,  0.]]]),
 'DESCR': ".. _digits_dataset:\n\nOptical recognition of handwritten digits dataset\n--
----------------------------------------\n\n**Data Set Characte
```

ristics:**\n\n   :Number of Instances: 5620\n   :Number of Attributes: 64\n   :Attribute Information: 8x8 image of integer pixels in the range 0..16.\n   :Missing Attribute Values: None\n   :Creator: E. Alpaydin (alpaydin '@' boun.edu.tr)\n   :Date: July; 1998\n\nThis is a copy of the test set of the UCI ML hand-written digits datasets\nhttp://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits\n\nThe data set contains images of hand-written digits: 10 classes where\neach class refers to a digit.\n\nPreprocessing programs made available by NIST were used to extract\nnormalized bitmaps of handwritten digits from a preprinted form. From a\ntotal of 43 people, 30 contributed to the training set and different 13\nto the test set. 32x32 bitmaps are divided into nonoverlapping blocks of\n4x4 and the number of on pixels are counted in each block. This generates\nan input matrix of 8x8 where each element is an integer in the range\n0..16. This reduces dimensionality and gives invariance to small\ndistortions.\n\nFor info on NIST preprocessing routines, see M. D. Garris, J. L. Blue, G.\nT. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Janet, and C.\nL. Wilson, NIST Form-Based Handprint Recognition System, NISTIR 5469,\n1994.\n\n.. topic:: References\n\n - C. Kaynak (1995) Methods of Combining Multiple Classifiers and Their\n   Applications to Handwritten Digit Recognition, MSc Thesis, Institute of\n   Graduate Studies in Science and Engineering, Bogazici University.\n - E. Alpaydin, C. Kaynak (1998) Cascading Classifiers, Kybernetika.\n - Ken Tang and Ponnuthurai N. Suganthan and Xi Yao and A. Kai Qin.\n   Linear dimensionalityreduction using relevance weighted LDA. School of\n   Electrical and Electronic Engineering Nanyang Technological University.\n   2005.\n - Claudio Gentile. A New Approximate Maximal Margin Classification\n   Algorithm. NIPS. 2000."}

In [5]:

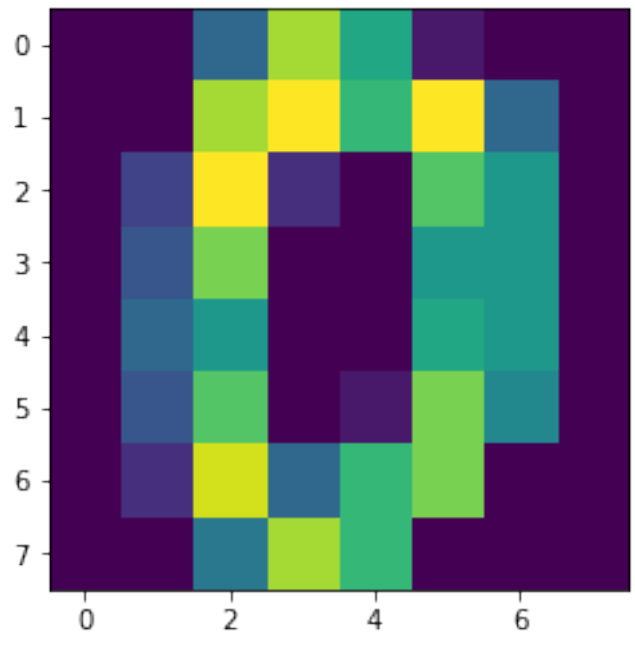
```
train = digits.data
target = digits.target
images = digits.images
```

In [6]:

```
import matplotlib.pyplot as plt
%matplotlib inline

# plt.imshow(images[0])
plt.imshow(train[0].reshape(8,8))
```

Out[6]:

<matplotlib.image.AxesImage at 0x112b40dd8>



In [7]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(train,target)
```

导入数据load_digits()

In [8]:

```
from sklearn.linear_model import LogisticRegression
```

创建模型，训练和预测

In [9]:

```
logistic = LogisticRegression(C=0.1)

# C惩罚系数 允许误差的阈值
# C越大，允许的误差越大
logistic.fit(X_train,y_train)

y_ = logistic.predict(X_test)

logistic.score(X_test,y_test)
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/skl
earn/linear_model/logistic.py:432: FutureWarning: Default solver will be changed to 'lbf
gs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/skl
earn/linear_model/logistic.py:459: FutureWarning: Default multi_class will be changed t
o 'auto' in 0.22. Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

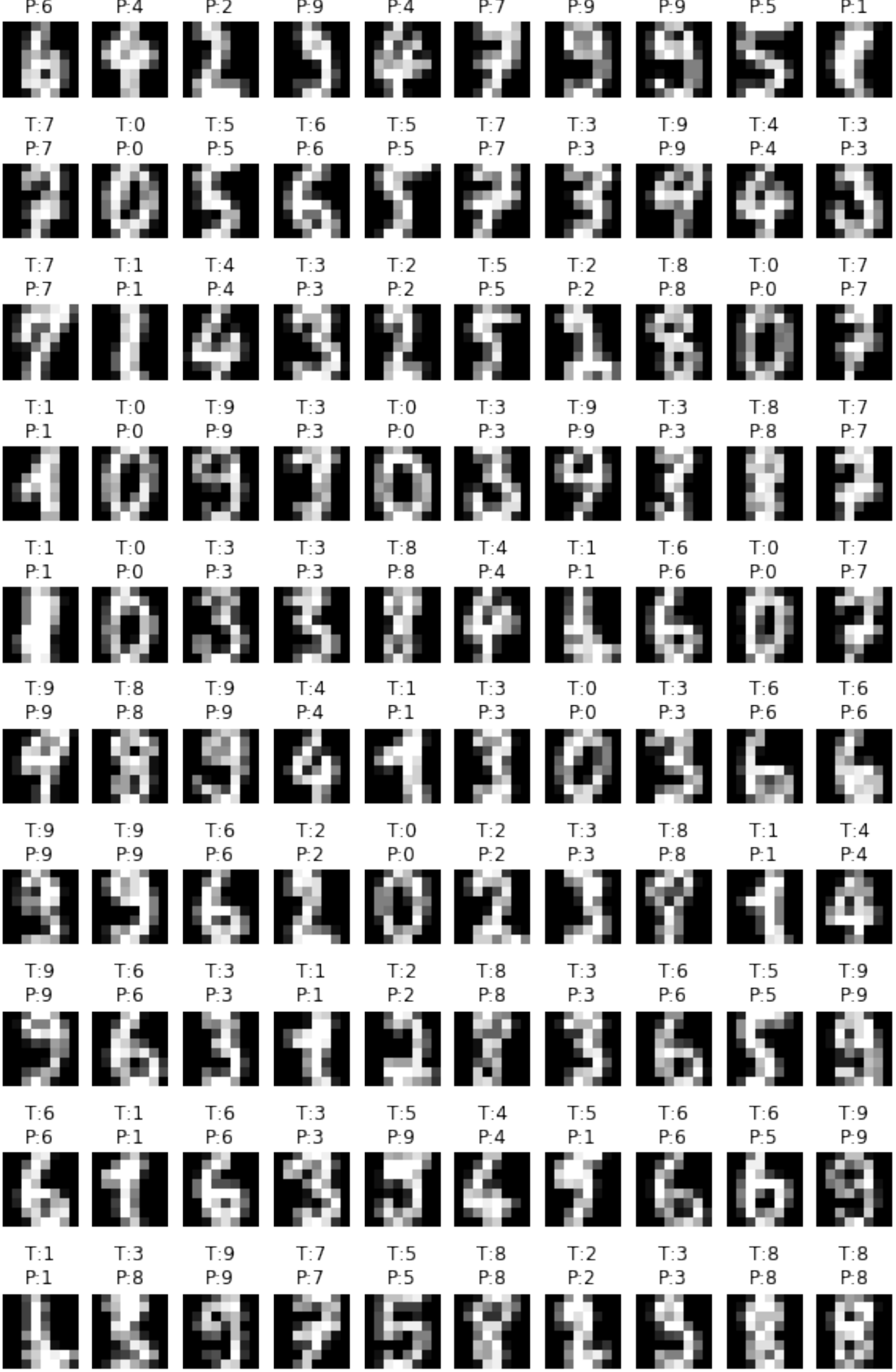
Out[9]:

0.9755555555555555

展示结果

In [10]:

```
plt.figure(figsize=(10,16))

for i in range(100):
    axes = plt.subplot(10,10,i+1)
    data = X_test[i].reshape(8,8)
    plt.imshow(data,cmap='gray')
    t = y_test[i]
    p = y_[i]
    title = 'T:'+str(t) + '\nP:'+str(p)
    axes.set_title(title)
    axes.axis('off')
```



2) 使用make_blobs产生数据集进行分类

导包使用datasets.make_blobs创建一系列点

In [41]:

```
from sklearn.datasets import make_blobs
from sklearn.neighbors import KNeighborsClassifier

import numpy as np
import pandas as pd
```

设置三个中心点，随机创建100个点

In [26]:

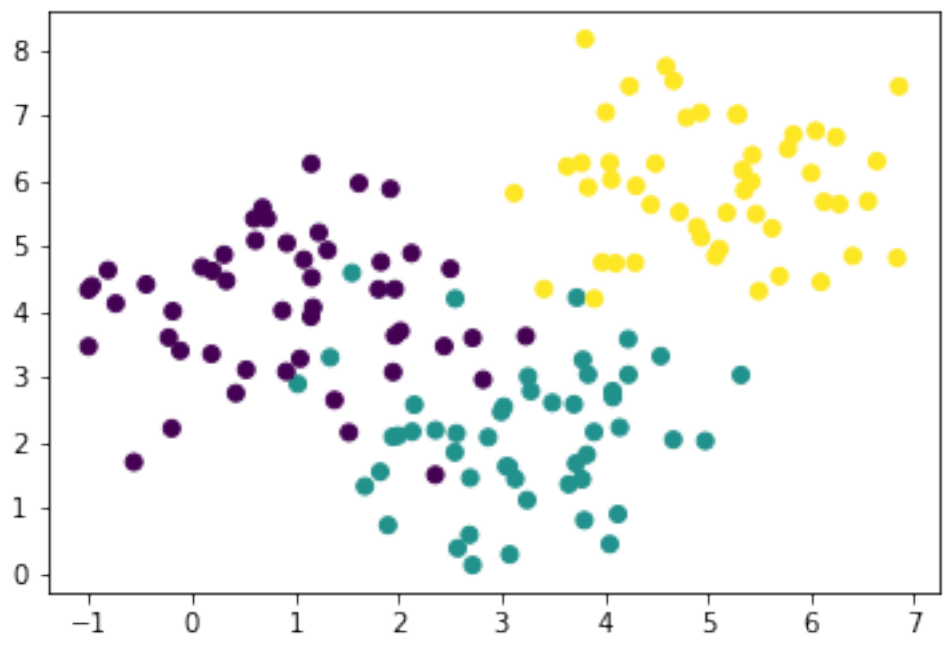
```
train,target = make_blobs(n_samples=150,n_features=2,centers=[[1,4],[3,2],[5,6]])
```

In [40]:

```
plt.scatter(train[:,0],train[:,1],c=target)
```

Out[40]:

<matplotlib.collections.PathCollection at 0x1796dc50>



创建机器学习模型，训练数据

In [42]:

```
logistic = LogisticRegression()
knnclf = KNeighborsClassifier()

logistic.fit(train,target)
knnclf.fit(train,target)
```

. . .

提取坐标点，对坐标点进行处理

In [44]:

```
# 获取边界
xmin,xmax = train[:,0].min()-0.5, train[:,1].max()+0.5
ymin,ymax = train[:,1].min()-0.5, train[:,1].max()+0.5
```

In [45]:

```
# 等差数列
x = np.linspace(xmin,xmax,200)
y = np.linspace(ymin,ymax,200)
```

In [46]:

```
# x和y交叉
xx,yy = np.meshgrid(x,y)
```

In [48]:

```
X_test = np.c_[xx.ravel(),yy.ravel()]
```

In [64]:

```
X_test.shape
```

Out[64]:

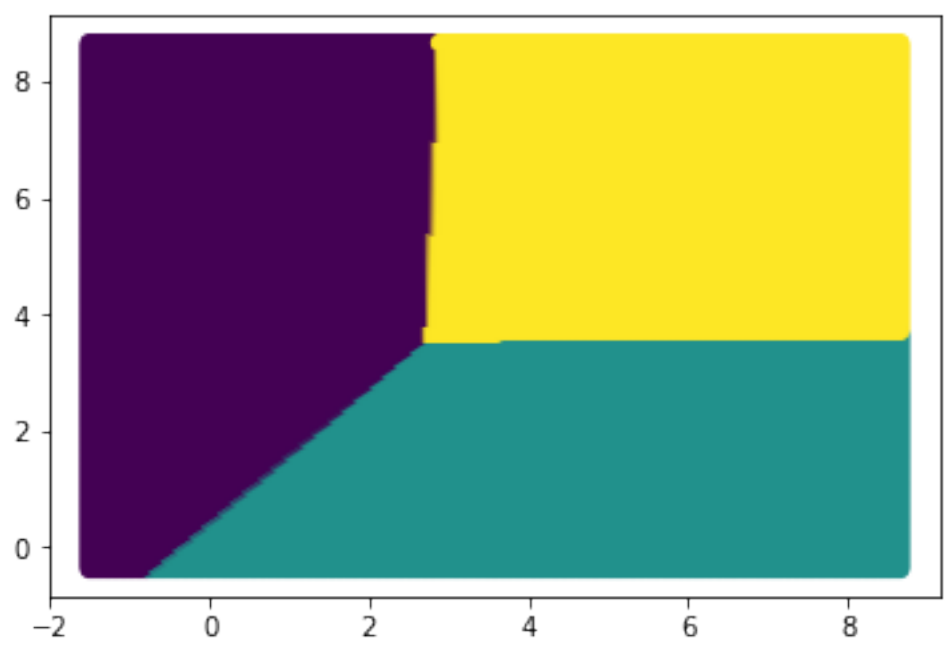
(40000, 2)

In [65]:

```

```

Out[65]:

<matplotlib.collections.PathCollection at 0x1bfcddd0>



预测坐标点数据，并进行reshape()

In [49]:

```
%time y1_ = logistic.predict(X_test)
```

Wall time: 3.97 ms

In [50]:

```
%time y2_ = knnclf.predict(X_test)
```

Wall time: 87.2 ms

绘制图形

In [55]:

```
from matplotlib.colors import ListedColormap
```

In [70]:

```
colormap = ListedColormap(['#aa00ff','#00aaff','#aaffff'])

def draw_classifier_bounds(X_train,y_train,X_test,y_test):
    plt.figure(figsize=(10,8))
    axes = plt.subplot(111)
    axes.scatter(X_test[:,0],X_test[:,1],c=y_test,cmap=colormap)
    axes.scatter(X_train[:,0],X_train[:,1],c=y_train)
```

In [71]:

```
draw_classifier_bounds(train,target,X_test,y1_)
```

. . .

In [72]:

```
draw_classifier_bounds(train,target,X_test,y2_)
```

. . .

3、作业

【第1题】预测年收入是否大于50K美元

读取adult.txt文件，并使用逻辑斯底回归算法训练模型，根据种族、职业、工作时长来预测一个人的性别

In [74]:

```
samples = pd.read_csv('../data/adults.txt')
```

In [75]:

```
samples.head(2)
```

Out[75]:

	age	workclass	final_weight	education	education_num	marital_status	occupation	relatic
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	N
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Hu

In [99]:

```
train = samples[['race','occupation','hours_per_week']].copy()
target = samples['sex']
```

In [100]:

```
train['race'].unique()
```

Out[100]:

```
array(['White', 'Black', 'Asian-Pac-Islander', 'Amer-Indian-Eskimo',
       'Other'], dtype=object)
```

In [101]:

```
race_dic = {
    'White':0,
    'Black':1,
    'Asian-Pac-Islander':2,
    'Amer-Indian-Eskimo':3,
    'Other':4
}
```

In [102]:

```
train['race'] = train['race'].map(race_dic)
```

In [104]:

```
unique_arr = train['occupation'].unique()
def transform_occ(x):
    return np.argwhere(x == unique_arr)[0,0]
```

In [105]:

```
train['occupation'] = train['occupation'].map(transform_occ)
```

In [106]:

```
train
```

. . .

In [92]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
```

In [107]:

```
X_train,X_test,y_train,y_test = train_test_split(train,target,test_size=0.2,random_state=1)
```

In [117]:

```
logistic = LogisticRegression(C=100)
knnclf = KNeighborsClassifier(n_neighbors=9)

logistic.fit(X_train,y_train)
knnclf.fit(X_train,y_train)

y1_ = logistic.predict(X_test)
y2_ = knnclf.predict(X_test)

print('logistic score is %f'%logistic.score(X_test,y_test))
print('knnclf score is %f'%knnclf.score(X_test,y_test))
```

```
logistic score is 0.681406
knnclf score is 0.714417
```

In [121]:

```
# 由于评分较低，把所有的数据特征都保留

train = samples.drop('sex',axis=1).copy()
target = samples.sex
```

In [123]:

```
train.head(2)
```

Out[123]:

	age	workclass	final_weight	education	education_num	marital_status	occupation	relatic
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	N
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Hu

In [129]:

```
columns = train.columns[train.dtypes == object]
for column in columns:
    unique_arr = train[column].unique()
    def transform_obj(x):
        return np.argwhere(x == unique_arr)[0,0]
    train[column] = train[column].map(transform_obj)
```

In [130]:

```
train.dtypes
```

```
. . .
```

In [131]:

```
X_train,X_test,y_train,y_test = train_test_split(train,target,test_size=0.2,random_state=1)
```

In [138]:

```
logistic = LogisticRegression(C=0.01)
knnclf = KNeighborsClassifier(n_neighbors=5)

logistic.fit(X_train,y_train)
knnclf.fit(X_train,y_train)

y1_ = logistic.predict(X_test)
y2_ = knnclf.predict(X_test)

print('logistic score is %f'%logistic.score(X_test,y_test))
print('knnclf score is %f'%knnclf.score(X_test,y_test))
```

```
logistic score is 0.668356
knnclf score is 0.667895
```

【第2题】从疝气病症预测病马的死亡率

逻辑斯蒂回归

【关键词】Logistics函数，最大似然估计，梯度下降法

1、Logistics回归的原理

利用Logistics回归进行分类的主要思想是：根据现有数据对分类边界线建立回归公式，以此进行分类。这里的“回归” 一词源于最佳拟合，表示要找到最佳拟合参数集。

训练分类器时的做法就是寻找最佳拟合参数，使用的是最优化算法。接下来介绍这个二值型输出分类器的数学原理

Logistic Regression和Linear Regression的原理是相似的，可以简单的描述为这样的过程：

（1）找一个合适的预测函数，一般表示为h函数，该函数就是我们需要找的分类函数，它用来预测输入数据的判断结果。这个过程是非常关键的，需要对数据有一定的了解或分析，知道或者猜测预测函数的“大概”形式，比如是线性函数还是非线性函数。

（2）构造一个Cost函数（损失函数），该函数表示预测的输出（h）与训练数据类别（y）之间的偏差，可以是二者之间的差（h-y）或者是其他的形式。综合考虑所有训练数据的“损失”，将Cost求和或者求平均，记为J(θ)函数，表示所有训练数据预测值与实际类别的偏差。

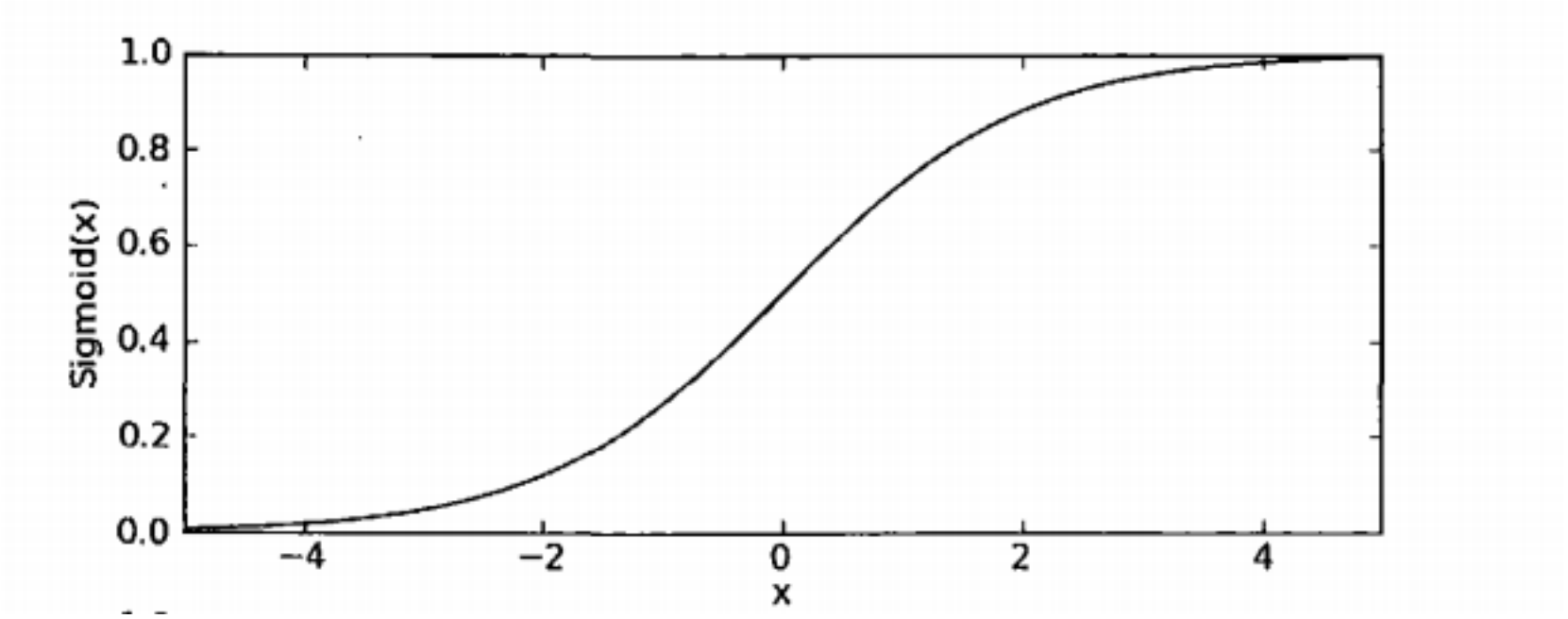
（3）显然，J(θ)函数的值越小表示预测函数越准确（即h函数越准确），所以这一步需要做的是找到J(θ)函数的最小值。找函数的最小值有不同的方法，Logistic Regression实现时有梯度下降法（Gradient Descent）。

1) 构造预测函数

Logistic Regression虽然名字里带“回归”，但是它实际上是一种分类方法，用于两分类问题（即输出只有两种）。首先需要先找到一个预测函数（h），显然，该函数的输出必须是两类值（分别代表两个类别），所以利用了*Logistic函数（或称为Sigmoid函数）*，函数形式为：

$$\sigma(z)=\frac{1}{1+e^{-z}}$$

该函数形状为：



预测函数可以写为：

$$h_\theta(x)=g(\theta^T x)=\frac{1}{1+e^{-\theta^T x}} \quad (3)$$

2）构造损失函数

Cost函数和J(θ)函数是基于*最大似然估计*推导得到的。

每个样本属于其真实标记的概率，即似然函数，可以写成：

$$P(y \mid x;\theta) = (h_\theta(x))^{y}(1-h_\theta(x))^{1-y} \tag{7}$$

所有样本都属于其真实标记的概率为

$$L(\theta) = \prod_{i=1}^{m} P(y^{(i)} \mid x^{(i)};\theta)$$

$$= \prod_{i=1}^{m} (h_\theta(x^{(i)}))^{y^{(i)}} (1-h_\theta(x^{(i)}))^{1-y^{(i)}} \tag{8}$$

对数似然函数为

$$l(\theta) = \log L(\theta)$$

$$= \sum_{i=1}^{m} \left(y^{(i)} \log h_\theta(x^{(i)}) + \left(1-y^{(i)}\right) \log(1-h_\theta(x^{(i)})) \right) \tag{9}$$

最大似然估计就是要求得使l(θ)取最大值时的θ，其实这里可以使用梯度上升法求解，求得的θ就是要求的最佳参数

3) 梯度下降法求J(θ)的最小值

求J(θ)的最小值可以使用*梯度下降法*，根据梯度下降法可得θ的更新过程:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta), \quad (j = 0 \ldots n) \tag{11}$$

式中为α学习步长，下面来求偏导：

$$\frac{\partial}{\partial\theta_j}J(\theta)=-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\frac{1}{h_\theta(\mathrm{x}^{(\mathrm{i})})}\frac{\partial}{\partial\theta_j}h_\theta(\mathrm{x}^{(\mathrm{i})})-(1-y^{(i)})\frac{1}{1-h_\theta(\mathrm{x}^{(\mathrm{i})})}\frac{\partial}{\partial\theta_j}h_\theta(\mathrm{x}^{(\mathrm{i})})\right)$$

$$=-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\frac{1}{g(\theta^T x^{(\mathrm{i})})}-(1-y^{(i)})\frac{1}{1-g(\theta^T x^{(\mathrm{i})})}\right)\frac{\partial}{\partial\theta_j}g(\theta^T x^{(\mathrm{i})})$$

$$=-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\frac{1}{g(\theta^T x^{(\mathrm{i})})}-(1-y^{(i)})\frac{1}{1-g(\theta^T x^{(\mathrm{i})})}\right)g(\theta^T x^{(\mathrm{i})})\left(1-g(\theta^T x^{(\mathrm{i})})\right)\frac{\partial}{\partial\theta_j}\theta^T x^{(\mathrm{i})}$$

$$=-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\left(1-g(\theta^T x^{(\mathrm{i})})\right)-(1-y^{(i)})g(\theta^T x^{(\mathrm{i})})\right)x_j^{(\mathrm{i})}$$

$$=-\frac{1}{m}\sum_{i=1}^{m}\left(y^{i}-g(\theta^T x^{(\mathrm{i})})\right)x_j^{(\mathrm{i})}$$

$$=-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}-h_\theta(\mathrm{x}^{(\mathrm{i})})\right)x_j^{(\mathrm{i})}$$

$$=\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(\mathrm{x}^{(\mathrm{i})})-y^{(i)}\right)x_j^{(\mathrm{i})}$$

（12）

上式求解过程中用到如下的公式：

$$f(x)=\frac{1}{1+e^{g(\mathrm{x})}}$$

$$\frac{\partial}{\partial x}f(x)=\frac{1}{\left(1+e^{g(\mathrm{x})}\right)^2}e^{g(\mathrm{x})}\frac{\partial}{\partial x}g(x)$$

（13）

$$=\frac{1}{1+e^{g(\mathrm{x})}}\frac{e^{g(\mathrm{x})}}{1+e^{g(\mathrm{x})}}\frac{\partial}{\partial x}g(x)$$

$$=f(x)\left(1-f(x)\right)\frac{\partial}{\partial x}g(x)$$

因此，θ的更新过程可以写成:

$$\theta_j:=\theta_j-\alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(\mathrm{x}^{(\mathrm{i})})-y^{(i)}\right)x_j^{(\mathrm{i})},\quad(j=0\ldots n)\qquad(14)$$

因为式中α本来为一常量，所以1/m一般将省略，所以最终的θ更新过程为：

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} \left(h_\theta(\mathrm{x}^{(i)}) - y^{(i)} \right) x_j^{(i)}, \quad (j = 0 \ldots n) \qquad (15)$$

Type *Markdown* and LaTeX: α^2

2、实战

sklearn.linear_model.LogisticRegression(penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='liblinear', max_iter=100, multi_class='ovr', verbose=0, warm_start=False, n_jobs=1)

solver参数的选择：

- “liblinear”：小数量级的数据集
- “lbfgs”, “sag” or “newton-cg”：大数量级的数据集以及多分类问题
- “sag”：极大的数据集

In [1]:

1) 手写数字数据集的分类

使用KNN与Logistic回归两种方法

In [2]:

In [3]:

In [4]:

Out[4]:

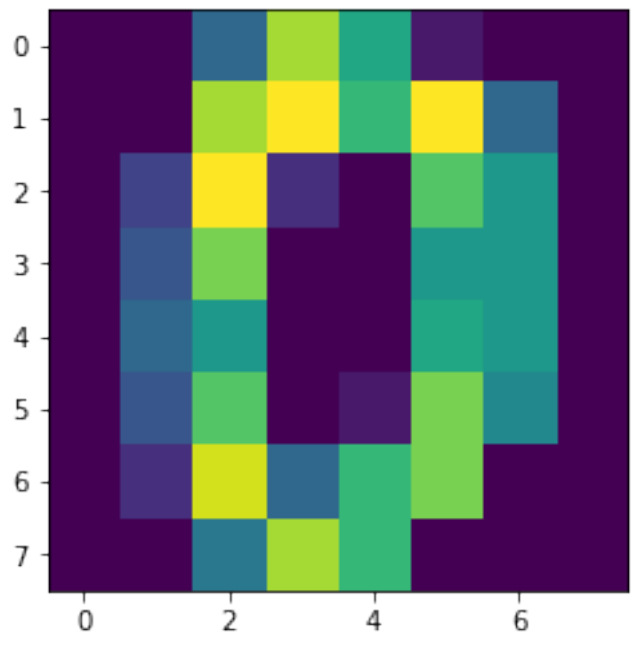
```
{'data': array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
        [ 0.,  0.,  0., ..., 10.,  0.,  0.],
        [ 0.,  0.,  0., ..., 16.,  9.,  0.],
        ...,
        [ 0.,  0.,  1., ...,  6.,  0.,  0.],
        [ 0.,  0.,  2., ..., 12.,  0.,  0.],
        [ 0.,  0., 10., ..., 12.,  1.,  0.]]),
 'target': array([0, 1, 2, ..., 8, 9, 8]),
 'target_names': array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 'images': array([[[ 0.,  0.,  5., ...,  1.,  0.,  0.],
         [ 0.,  0., 13., ..., 15.,  5.,  0.],
         [ 0.,  3., 15., ..., 11.,  8.,  0.],
         ...,
         [ 0.,  4., 11., ..., 12.,  7.,  0.],
         [ 0.,  2., 14., ..., 12.,  0.,  0.],
         [ 0.,  0.,  6., ...,  0.,  0.,  0.]],

        [[ 0.,  0.,  0., ...,  5.,  0.,  0.],
```

In [5]:

In [6]:

Out[6]:

<matplotlib.image.AxesImage at 0x112b40dd8>



In [7]:

导入数据load_digits()

In [8]:

创建模型，训练和预测

In [9]:

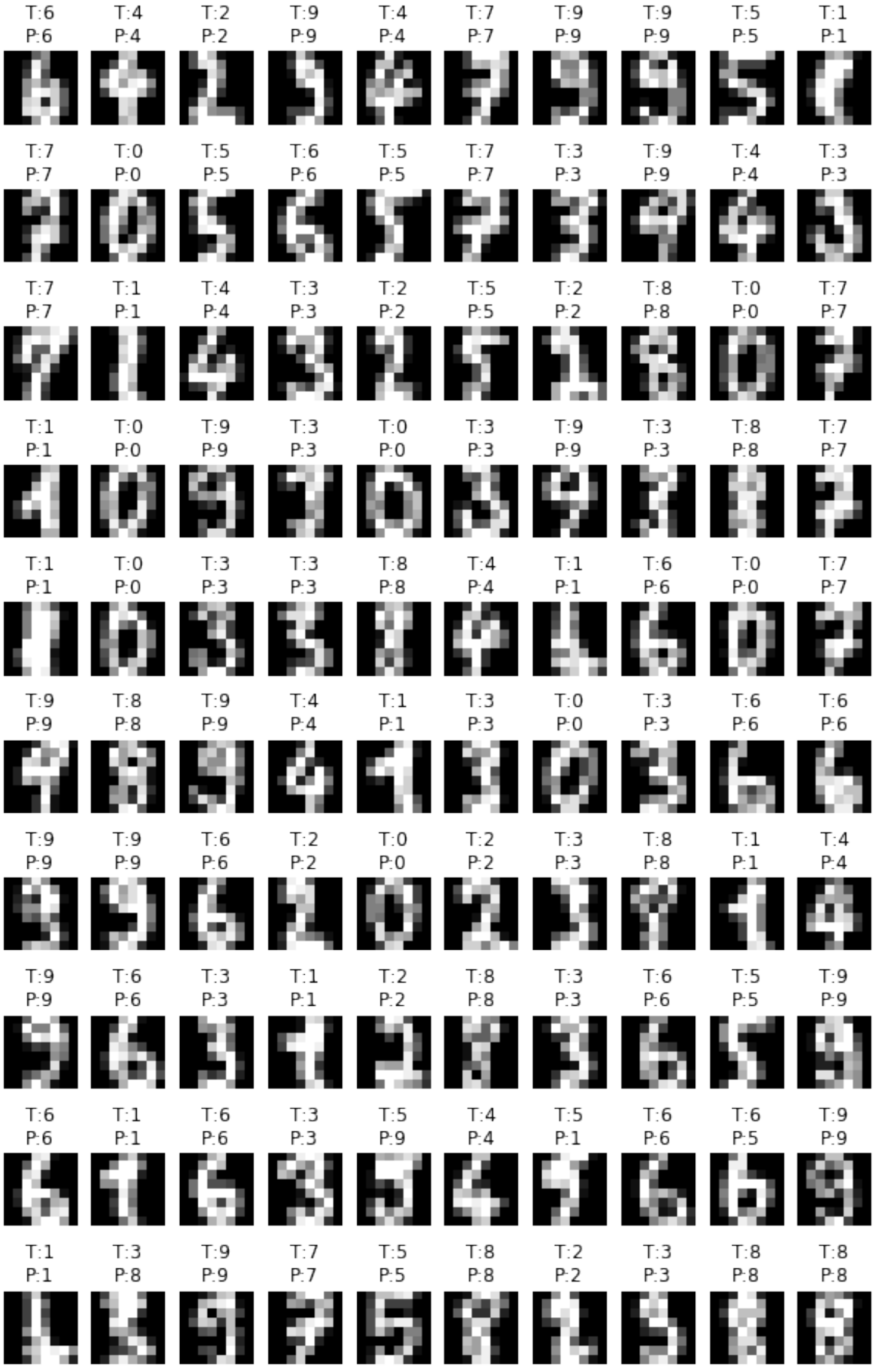
```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/skl
earn/linear_model/logistic.py:432: FutureWarning: Default solver will be changed to 'lbf
gs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/skl
earn/linear_model/logistic.py:459: FutureWarning: Default multi_class will be changed t
o 'auto' in 0.22. Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

Out[9]:

```
0.9755555555555555
```

展示结果

In [10]:


T:6 P:6
T:4 P:4
T:2 P:2
T:9 P:9
T:4 P:4
T:7 P:7
T:9 P:9
T:9 P:9
T:5 P:5
T:1 P:1
T:7 P:7
T:0 P:0
T:5 P:5
T:6 P:6
T:5 P:5
T:7 P:7
T:3 P:3
T:9 P:9
T:4 P:4
T:3 P:3
T:7 P:7
T:1 P:1
T:4 P:4
T:3 P:3
T:2 P:2
T:5 P:5
T:2 P:2
T:8 P:8
T:0 P:0
T:7 P:7
T:1 P:1
T:0 P:0
T:9 P:9
T:3 P:3
T:0 P:0
T:3 P:3
T:9 P:9
T:3 P:3
T:8 P:8
T:7 P:7
T:1 P:1
T:0 P:0
T:3 P:3
T:3 P:3
T:8 P:8
T:4 P:4
T:1 P:1
T:6 P:6
T:0 P:0
T:7 P:7
T:9 P:9
T:8 P:8
T:9 P:9
T:4 P:4
T:1 P:1
T:3 P:3
T:0 P:0
T:3 P:3
T:6 P:6
T:6 P:6
T:9 P:9
T:9 P:9
T:6 P:6
T:2 P:2
T:0 P:0
T:2 P:2
T:3 P:3
T:8 P:8
T:1 P:1
T:4 P:4
T:9 P:9
T:6 P:6
T:3 P:3
T:1 P:1
T:2 P:2
T:8 P:8
T:3 P:3
T:6 P:6
T:5 P:5
T:9 P:9
T:6 P:6
T:1 P:1
T:6 P:6
T:3 P:3
T:5 P:9
T:4 P:4
T:5 P:1
T:6 P:6
T:6 P:5
T:9 P:9
T:1 P:1
T:3 P:8
T:9 P:9
T:7 P:7
T:5 P:5
T:8 P:8
T:2 P:2
T:3 P:3
T:8 P:8
T:8 P:8

2) 使用make_blobs产生数据集进行分类

导包使用datasets.make_blobs创建一系列点

In [41]:

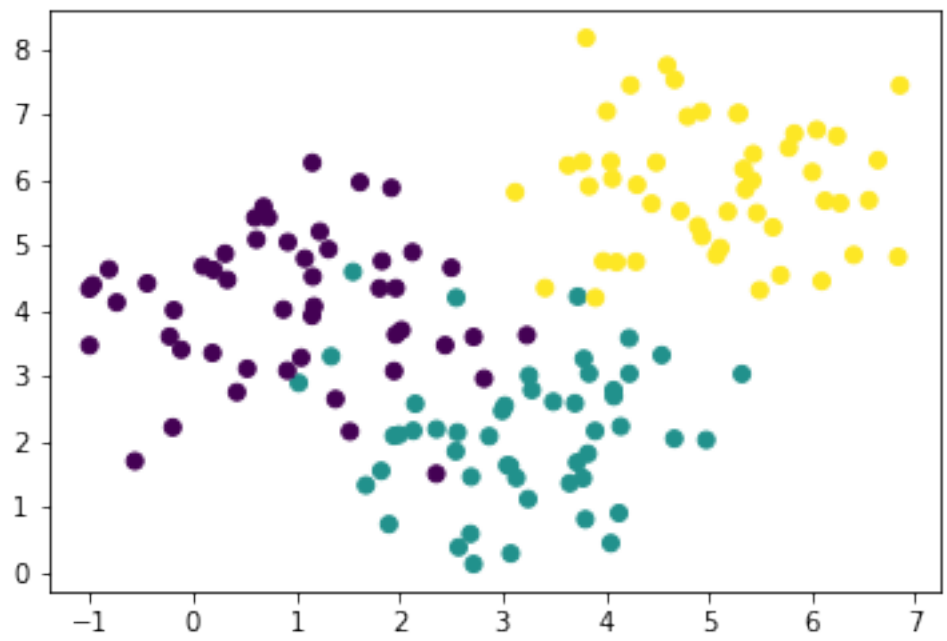
设置三个中心点，随机创建100个点

In [26]:

In [40]:

Out[40]:

<matplotlib.collections.PathCollection at 0x1796dc50>



创建机器学习模型，训练数据

In [42]:

...

提取坐标点，对坐标点进行处理

In [44]:

In [45]:

In [46]:

In [48]:

In [64]:

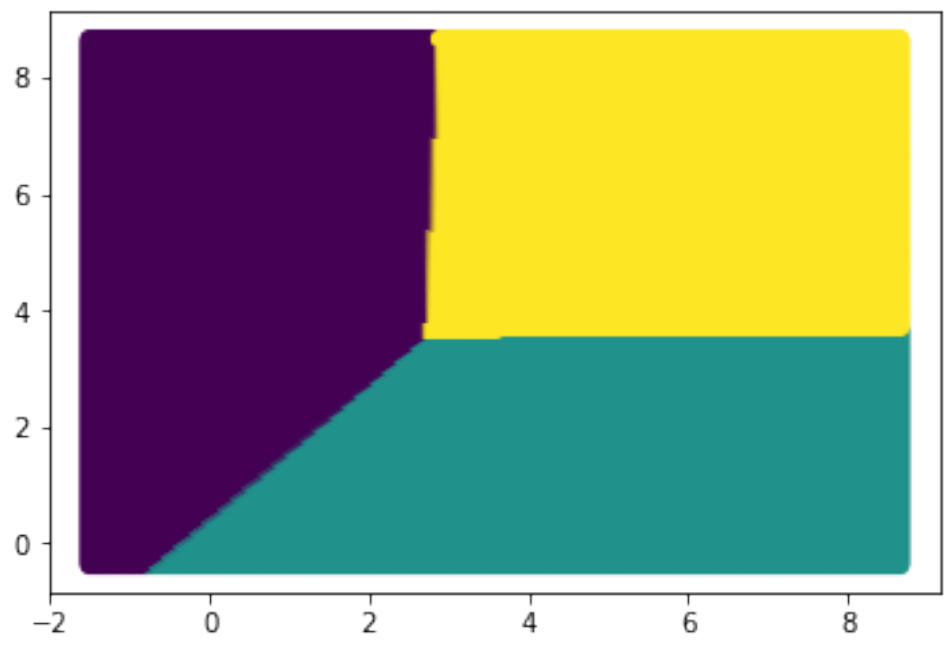
Out[64]:

(40000, 2)

In [65]:

Out[65]:

<matplotlib.collections.PathCollection at 0x1bfcddd0>



预测坐标点数据，并进行reshape()

In [49]:

Wall time: 3.97 ms

In [50]:

Wall time: 87.2 ms

绘制图形

In [55]:

In [70]:

In [71]:

```
...
```

In [72]:

```
...
```

3、作业

【第1题】预测年收入是否大于50K美元

读取adult.txt文件，并使用逻辑斯底回归算法训练模型，根据种族、职业、工作时长来预测一个人的性别

In [74]:

In [75]:

Out[75]:

	age	workclass	final_weight	education	education_num	marital_status	occupation	relatic
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	N
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Hu

In [99]:

In [100]:

Out[100]:

```
array(['White', 'Black', 'Asian-Pac-Islander', 'Amer-Indian-Eskimo',
       'Other'], dtype=object)
```

In [101]:

In [102]:

In [104]:

In [105]:

In [106]:

```
...
```

In [92]:

In [107]:

In [117]:

```
logistic score is 0.681406
knnclf score is 0.714417
```

In [121]:

In [123]:

Out[123]:

	age	workclass	final_weight	education	education_num	marital_status	occupation	relatic
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	N
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Hu

In [129]:

In [130]:

```
...
```

In [131]:

In [138]:

```
logistic score is 0.668356
knnclf score is 0.667895
```

【第2题】从疝气病症预测病马的死亡率