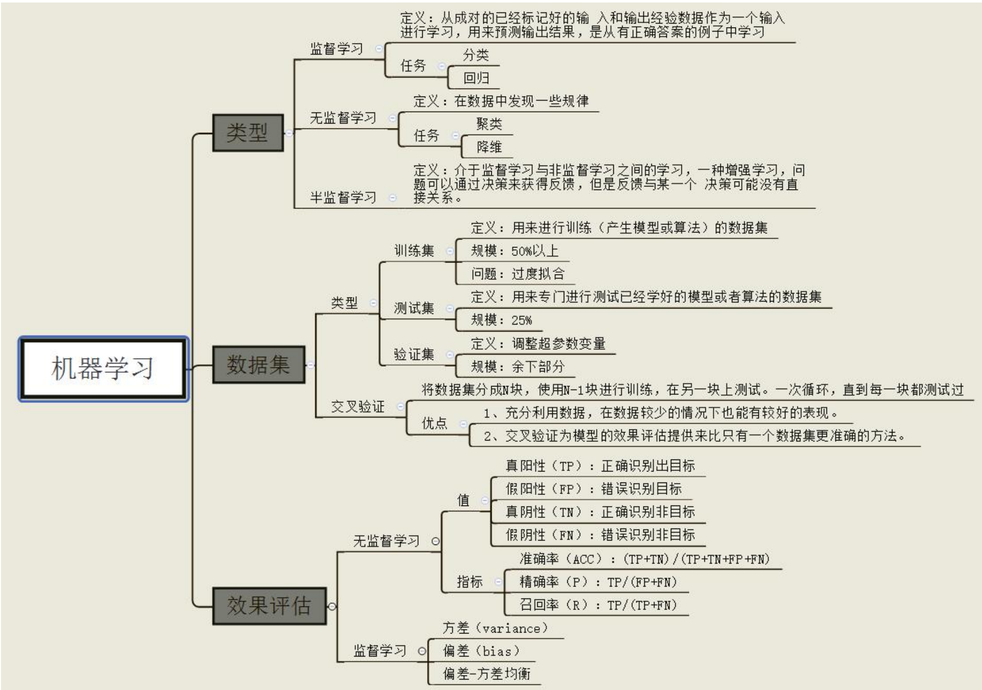
学习 sklearn 库——WittPeng（Using Python 2.7）

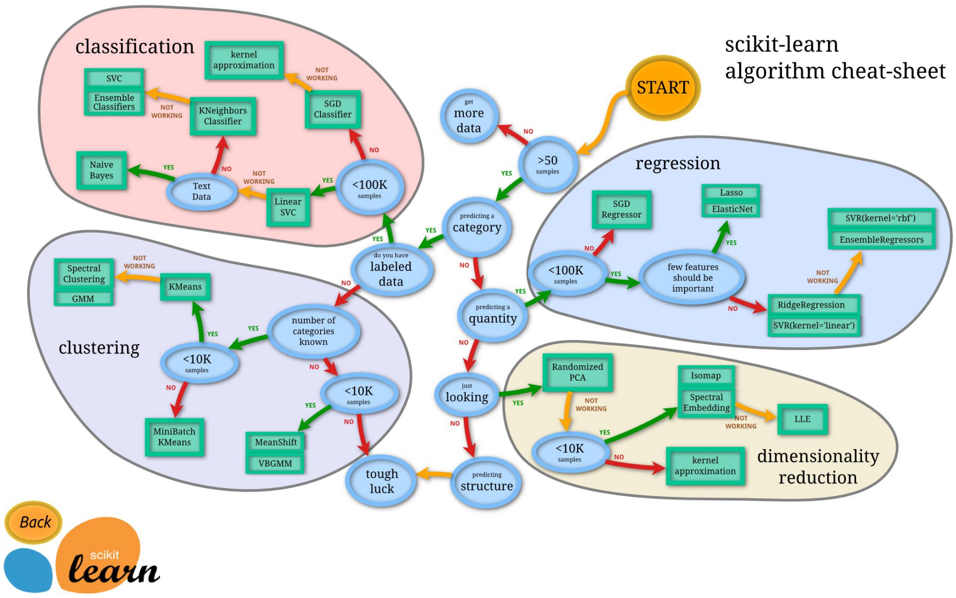
机器学习的概念 https://www.jianshu.com/p/28f02bb59fe5

机器学习的初衷就是希望计算机像人一样思考，可行性在于计算机和人一样都是由最基本的单元组成起来的。机器学习应用广泛，谷歌是杰出的使用机器学习技术的企业代表之一。机器学习的算法多种多样，不同的算法就是对数据不同的处理思想和方法。如：监督学习、无监督学习、半监督学习、强化学习和遗传算法等，机器学习的结构和内容为：



sklearn

它的结构是这样的：



（1）结构：

由图中，可以看到库的算法主要有四类：分类，回归，聚类，降维。其中：

常用的回归：线性、决策树、SVM、KNN ；集成回归：随机森林、Adaboost、GradientBoosting、Bagging、ExtraTrees

常用的分类：线性、决策树、SVM、KNN，朴素贝叶斯；集成分类：随机森林、Adaboost、GradientBoosting、Bagging、ExtraTrees

常用聚类：k 均值（K-means）、层次聚类（Hierarchical clustering）、DBSCAN

常用降维：LinearDiscriminantAnalysis、PCA

（2）图片中隐含的操作流程：

这个流程图代表：蓝色圆圈内是判断条件，绿色方框内是可以选择的算法。

I 入门推荐参考：https://morvanzhou.github.io/tutorials/machine-learning/sklearn/

1.KNN 的初步使用

使用 scikit-learn 自带的数据集 iris，IDA 环境为 Python 2.7

初始代码：

```
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

iris=datasets.load_iris()
iris_X=iris.data
iris_y=iris.target

print iris_X[:2,:]
```

输出：

```
(python27) zhanglipengdeMacBook-Pro:Desktop zhanglipeng$ python Knn2.py
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]]
```

这是 iris 数据集的两个实例，可以看每一个 sample 都有 4 个属性。

```
print iris_y
```

输出：

```
(python27) zhanglipengdeMacBook-Pro:Desktop zhanglipeng$ python Knn2.py
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

可以看到，一共有三种分类，分别是 0、1、2。

上网查看一下，iris 的数据集内容是这样的：

费雪鸢尾花卉数据集

花萼长度	花萼宽度	花瓣长度	花瓣宽度	属种
5.1	3.5	1.4	0.2	*setosa*
4.9	3.0	1.4	0.2	*setosa*
4.7	3.2	1.3	0.2	*setosa*
4.6	3.1	1.5	0.2	*setosa*
5.0	3.6	1.4	0.2	*setosa*
5.4	3.9	1.7	0.4	*setosa*
4.6	3.4	1.4	0.3	*setosa*
5.0	3.4	1.5	0.2	*setosa*

然后使用 import train_test_split 模块将训练集和测试集分开：

```
X_train,X_test,y_train,y_test=train_test_split(iris_X,iris_y,test_size=0.3)#split X and y,the testing proportion is 30%
print y_train
```

y_train 的查看结果为

```
(python27) MacBook-Pro:Desktop zhanglipeng$ python Knn2.py
[2 2 1 0 2 1 2 2 1 1 1 2 1 1 2 2 0 2 0 0 0 2 0 2 1 2 1 0 2 0 0 0 1 0 1 2 2
 2 2 2 2 2 0 0 1 2 2 0 0 2 0 0 0 1 0 1 2 0 2 0 0 1 2 2 1 0 0 2 2 0 1 1 1 1
 1 0 1 1 0 1 2 0 1 2 0 1 2 1 1 1 1 2 2 1 1 1 0 0 0 1 1 2 1 1 2]
```

可以看到，对数据进行了打乱，防止学习过程中出现了影响，随机分布更遂人意。

```
knn=KNeighborsClassifier()
knn.fit(X_train,y_train)
print knn.predict(X_test)
print y_test
```

输出结果：

```
(python27) zhanglipengdeMacBook-Pro:Desktop zhanglipeng$ python Knn2.py
[1 2 2 0 1 2 1 1 2 2 0 1 0 0 0 1 1 2 1 2 2 0 0 1 1 1 2 1 0 0 1 0 0 0 2 1 2
 1 2 0 0 0 0 1 2]
[1 2 2 0 1 2 1 1 2 2 0 1 0 0 0 1 1 2 1 2 2 0 0 1 1 1 2 1 0 0 1 0 0 0 2 1 2
 1 1 0 0 0 0 1 2]
```

可以看到，预测的结果大部分都是准确的。机器学习一般不会达到 100%，即使是人本身也不敢保证如此高的正确率。有的只能是更好的优化，更丰富的数据。

2.了解 sklearn 的数据集

Scikit-learn 的数据集非常丰富，用户还可以自己生成数据集。数据集查看网址。

使用方法：

```
from sklearn import datasets
from sklearn.linear_model import LinearRegression

loaded_data = datasets.load_boston()
data_X = loaded_data.data
data_y = loaded_data.target

model = LinearRegression()
model.fit(data_X,data_y)
```

```
print model.predict(data_X[:4,:])
print data_y[:4]
```

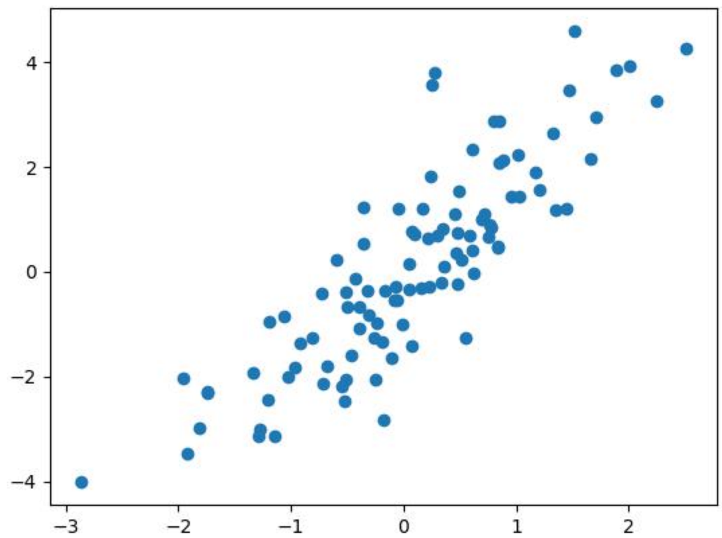
输出:

```
(python27) zhanglipengdeMacBook-Pro:Desktop zhanglipeng$ python Knn2.py
[30.00384338 25.02556238 30.56759672 28.60703649]
[24.   21.6 34.7 33.4]
```

创建属于自己的数据集:

```
from sklearn import datasets
from sklearn.linear_model import LinearRegression
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt

X,y=datasets.make_regression(n_samples=100,n_features=1,n_targets=1,noise=1)
plt.scatter(X,y)
plt.show()
```

输出:



3.sklearn 的 model 常用的属性和功能

```
from sklearn import datasets
from sklearn.linear_model import LinearRegression

loaded_data = datasets.load_boston()
data_X = loaded_data.data
data_y = loaded_data.target

model = LinearRegression()
```

```
model.fit(data_X,data_y)
#输出 y 和 x 的斜率和截距
print model.coef_
print model.intercept_
```

经过 fit 功能，训练完毕后的输出，结果较接近真实性。

输出：

```
(python27) zhanglipengdeMacBook-Pro:Desktop zhanglipeng$ python Knn2.py
[-1.08011358e-01  4.64204584e-02  2.05586264e-02  2.68673382e+00
 -1.77666112e+01  3.80986521e+00  6.92224640e-04 -1.47556685e+00
  3.06049479e-01 -1.23345939e-02 -9.52747232e-01  9.31168327e-03
 -5.24758378e-01]
36.459488385090125
```

[]里是数据集参数与其相乘，再相加。

```
print model.get_params()
```

输出：之前定义时设定参数的情况

```
(python27) zhanglipengdeMacBook-Pro:Desktop zhanglipeng$ python Knn2.py
{'copy_X': True, 'normalize': False, 'n_jobs': None, 'fit_intercept': True}
```

```
print model.score(data_X,data_y)#R^2coefficient of determination
```

输出：打分结果

```
(python27) zhanglipengdeMacBook-Pro:Desktop zhanglipeng$ python Knn2.py
0.7406426641094095
```

4.正规化操作（Normalization/Scale）

正规化操作的目的是为了将大跨度的数据处理一下，希望跨度能变成一个小范围，如[0，1]。

```
from sklearn import preprocessing
import numpy as np

a=np.array([ [10,2.7,3.6],
             [-100,5,-2],
             [120,20,40]],dtype=np.float64)
print a
print preprocessing.scale(a)
```

输出：

```
(python27) zhanglipengdeMacBook-Pro:Desktop zhanglipeng$ python Knn2.py
[[  10.     2.7    3.6]
 [-100.     5.    -2. ]
 [ 120.    20.    40. ]]
[[ 0.         -0.85170713 -0.55138018]
 [-1.22474487 -0.55187146 -0.852133  ]
 [ 1.22474487  1.40357859  1.40351318]]
```

```
from sklearn import preprocessing
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.datasets.samples_generator import make_classification
from sklearn.svm import SVC
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt

X,y=make_classification(n_samples=300,n_features=2,n_redundant=0,n_informative=2,random_state=22,
                        n_clusters_per_class=1,scale=100)
plt.scatter(X[:,0],X[:,1],c=y)

#X=preprocessing.scale(X)

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=3)
clf=SVC()
clf.fit(X_train,y_train)
print clf.score(X_test,y_test)
```

5.交叉验证（cross_validation）

用于判断此次训练中的参数效果如何。

初始代码:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

iris = load_iris()
X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=4)
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
print knn.score(X_test, y_test)
```

输出结果:

```
(python27) zhanglipengdeMacBook-Pro:Desktop zhanglipeng$ python Knn2.py
0.9736842105263158
```

使用交叉验证，就是把训练集逐一在原始数据集的范围内寻找，这样下来，我们可以把多组测试结果平均值拿出来，显得更加准确了。

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
```

```
iris = load_iris()
X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=4)
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred=knn.predict(X_test)
print knn.score(X_test, y_test)

from sklearn.model_selection import cross_val_score
scores = cross_val_score(knn, X, y, cv=5, scoring='accuracy')
print scores
print scores.mean()
```

输出：

```
(python27) zhanglipengdeMacBook-Pro:Desktop zhanglipeng$ python Knn2.py
0.9736842105263158
[0.96666667 1.         0.93333333 0.96666667 1.        ]
0.9733333333333334
```

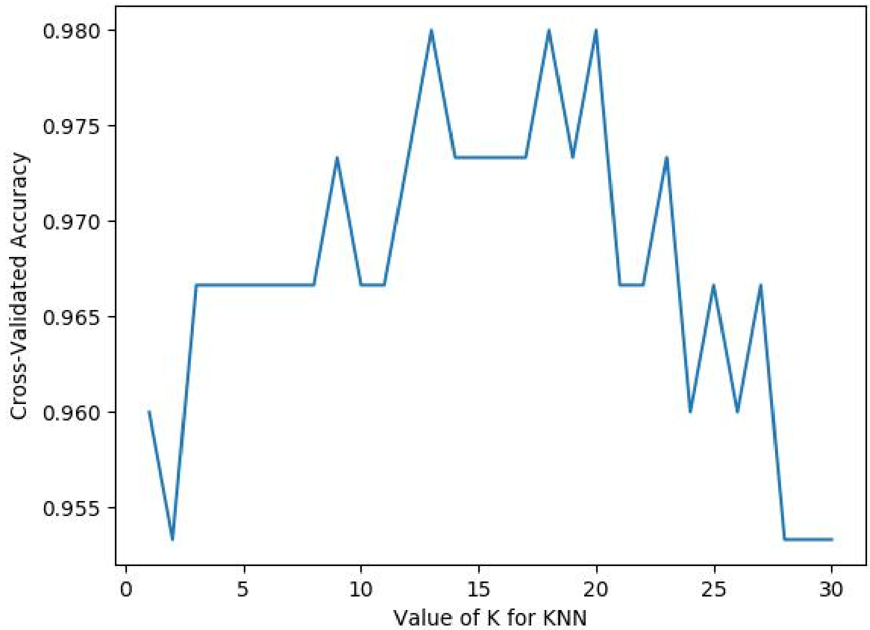
那么这时候，就会思考，在 knn 算法的使用中究竟多少邻居数合理呢？

代码：

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

iris = load_iris()
X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=4)
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred=knn.predict(X_test)
print knn.score(X_test, y_test)

from sklearn.model_selection import cross_val_score
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt


k_range = range(1, 31)
k_scores = []
```

```
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X, y, cv=10, scoring='accuracy')
    k_scores.append(scores.mean())

plt.plot(k_range, k_scores)
plt.xlabel('Value of K for KNN')
plt.ylabel('Cross-Validated Accuracy')
plt.show()
```

输出：



可以看到 k 值在 12-16 之间是比较好的，太多了反而会过度拟合。

也可以用 loss 率，进行选择。

说到过度拟合，这个问题可能造成分界界限不明确这样的问题，再比如生成一个近似为一元曲线，过度拟合则会造成曲线歪歪曲曲。

代码：

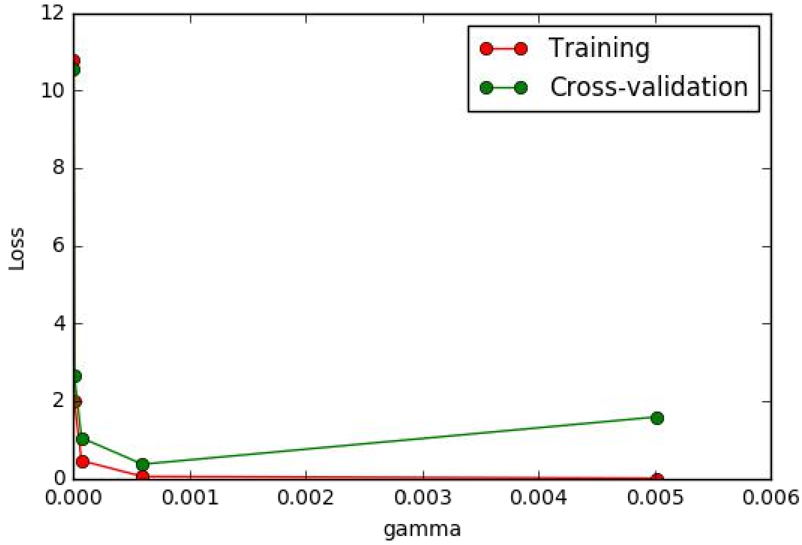
```
from sklearn.learning_curve import learning_curve
from sklearn.datasets import load_digits
from sklearn.svm import SVC
import matplotlib.pyplot as plt
import numpy as np
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt

digits = load_digits()
```

```
X = digits.data
y = digits.target

train_sizes, train_loss, test_loss = learning_curve(SVC(gamma=0.001), X, y, cv=10,scoring='mean_squared_error',train_sizes=[0.1, 0.25, 0.5, 0.75, 1])

train_loss_mean = -np.mean(train_loss, axis=1)
test_loss_mean = -np.mean(test_loss, axis=1)

plt.plot(train_sizes, train_loss_mean, 'o-', color="r",
         label="Training")
plt.plot(train_sizes, test_loss_mean, 'o-', color="g",
         label="Cross-validation")

plt.xlabel("Training examples")
plt.ylabel("Loss")
plt.legend(loc="best")
plt.show()
```

再说一下，为什么过拟合不好呢？因为训练样本有一定的片面性，过于贴合训练集的范围则对测试集反而不友好了。

```
from sklearn.model_selection import validation_curve
from sklearn.datasets import load_digits
from sklearn.svm import SVC
import numpy as np
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt

digits = load_digits()
X = digits.data
y = digits.target
param_range=np.logspace(-6,-2.3,5)
train_loss, test_loss = validation_curve(SVC(), X, y,param_name='gamma',param_range=param_range,cv=10,scoring='mean_squared_error')

train_loss_mean = -np.mean(train_loss, axis=1)
test_loss_mean = -np.mean(test_loss, axis=1)

plt.plot(param_range, train_loss_mean, 'o-', color="r",
         label="Training")
plt.plot(param_range, test_loss_mean, 'o-', color="g",
         label="Cross-validation")

plt.xlabel("gamma")
```

```
plt.ylabel("Loss")
plt.legend(loc="best")
plt.show()
```



可以看到，gamma 值在到达最低点后继续增大，就会出现过拟合的情况。

II.研习中文文档：http://cwiki.apachecn.org/pages/viewpage.action?pageId=10030181

https://www.jianshu.com/p/28f02bb59fe5