模型评估Model Evaluation

1. 性能评估矩阵(Metrics for Performance Evaluation)
2. 性能评估方法(Methods for Performance Evaluation)
3. 模型对比方法(Methods for Model Comparison)

1-性能评估矩阵(Metrics for Performance Evaluation)

关注的是模型的预测能力，而不是分类或构建模型的速度(Focus on the predictive capability of a model Rather than how fast it takes to classify or build models, scalability, etc.)

混淆矩阵(Confusion Matrix)

Count	Predicted Class		
Actaul Class		Class=Yes	Class=No
	Class=Yes	a:TP(True Positive)	b:FN(False Negative)
	Class=No	c:FP(False Postive)	d:TN(True Negative)

注:

1. TP(真阳性，真实类别为正例，预测为正例)
2. FN(假阴性，真实类别为正例，预测为负例)(误报)
3. FP(假阳性，真实类别为负例，预测为正例)(漏报)
4. TN(真隐性，真实类别为负例，预测为负例)

衡量指标:

1. 准确率(Accuracy)

$$Accuracy = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+FN+FP+TN}$$

当数据出现极度不均衡时，准确率会存在误导。如10000个样例9990个正例，10个负例，模型不需要学习将全部样本预测为正例得到的准确率为99.9%。

2. 精确率P(Precision)(查准率)

$$P(Precision) = \frac{a}{a+c} = \frac{TP}{TP+FP}$$

预测为正例的样本中，真的正例所占的比例

3. 召回率R(Recall)(查全率)

$$R(Recall) = \frac{a}{a+b} = \frac{TP}{TP+FN}$$

所有正实例中，预测出来的所占的比例

4. F因子(F-score,F-measure)

$$F - measure = \frac{(1+\beta^2)*P*R}{\beta^2 * P + R}$$

常见的有$F1 - measure = \frac{2*P*R}{P+R}$

Cost Matrix

		Predicted Class	
	C(i|j)	Class=Yes	Class=No
Actaul Class	Class=Yes	C(Yes|Yes) (TP)	C(No|Yes) (FN)
	Class=No	C(Yes|No) (FP)	C(No|No) (TN)

其中$C(i|j)$表示将类j预测为i的损失

$$C_t(M) = TP * C(Yes|Yes) + FN * C(No|Yes) + FP * C(Yes|No) + TN * C(No|No)$$

2-性能评估方法(Methods for Performance Evaluation)

How to obtain a reliable estimate of performance?

Performance of a model may depend on other factors besides the learning algorithm?

1. 类分布(Class distribution)
2. 错误分类损失(Cost of misclassification)
3. 训练集和测试集大小(Size of training and test sets)

Learning curve shows how accuracy changes with varying sample size. Requires a sampling schedule for creating learning curve:

1. Arithmetic sampling (Langley et al)
2. Geometric sampling (Provost et al)
3. Effect of small sample size
4. Bias in the estimate Variance of estimate

预留Holdout:

Reserve 2/3 for training and 1/3 for testing

Partition data set $D = \{(v_1, y_1), ..., (v_n, y_n)\}$ into training D_t and validation set $D_h = D - D_t$

Problem: 1. 尤其是对于数据比较少的情况下数据利用不充分(Makes insufficient use of data) 2. 训练数据和验证数据是相关(Training and validation set are correlated)

随机下采样Random subsampling:

```
repeated holdout
```

交叉验证Cross validation:

k-fold cross-validation splits the data set D into k mutually exclusive subsets $D_1, D_2, \cdots, D_k$

Train and test the learning algorithm k times, each time it is trained on $D - D_i$ and tested on D_i

如果$k = n$,留一交叉验证。常用的交叉验证方式如$5 * 10$折交叉验证和$10 * 10$折交叉雁阵。但是仍存在问题，如果数据集是时间序列等和时间有关的数据，采用该方法会产生使用时间上后产生的数据来预测时间上比较靠前的数据。

分层采样Stratified sampling

```
oversampling vs undersampling
```

Bootstrap:

有放回的从数据集中抽取n个实例,Samples n instances uniformly from the data set with replacement

Probability that any given instance is not chosen after n samples is

$$\left(1-\frac{1}{n}\right)^{n} \approx e^{-1} \approx 0.368$$

The bootstrap sample is used for training, the remaining instances are used for testing

$$acc_{boot} = \frac{1}{b}\sum_{i=1}^{b}(0.632\epsilon_{0i} + 0.368acc_s)$$

where ϵ_{0i} is the accuracy on the test data of the i-th bootstrap sample, acc_s is the accuracy estimate on the original set and b the number of bootstrap samples

```
#10折交叉验证 (10 flod cross validation)
def cross_10folds(data,n_splits=10,n_repeats=5):
    n_sample = data.shape[0]
    interval = n_sample//n_splits
    for i in range(n_repeats):
        indices = np.arange(n_sample)
        np.random.shuffle(indices)
        for j in range(n_splits):
            test_index = indices[j*interval:(j+1)*interval]
            train_index = np.append(indices[:j*interval],indices[(j+1)*interval:])
            yield train_index,test_index
```

```
import numpy as np
from sklearn.model_selection import KFold
data = np.array(["a", "b", "c", "d"])
kf = KFold(n_splits=2)
for train_index, test_index in kf.split(data):
    traindata,testdata = data[train_index],data[test_index]
    print(traindata,testdata)
```

```
['c' 'd'] ['a' 'b']
['a' 'b'] ['c' 'd']
```

```
import numpy as np
from sklearn.model_selection import RepeatedKFold
data = np.array([[1, 2], [3, 4], [1, 2], [3, 4]])
random_state = 12883823
rkf = RepeatedKFold(n_splits=2, n_repeats=1, random_state=random_state)
for train_index, test_index in rkf.split(data):
    print(train_index,test_index)
    traindata,testdata = data[train_index],data[test_index]
    #print(traindata,"\n",testdata)
for train_index, test_index in cross_10folds(data,n_splits=2, n_repeats=1):
    print(train_index,test_index)
    traindata,testdata = data[train_index],data[test_index]
```

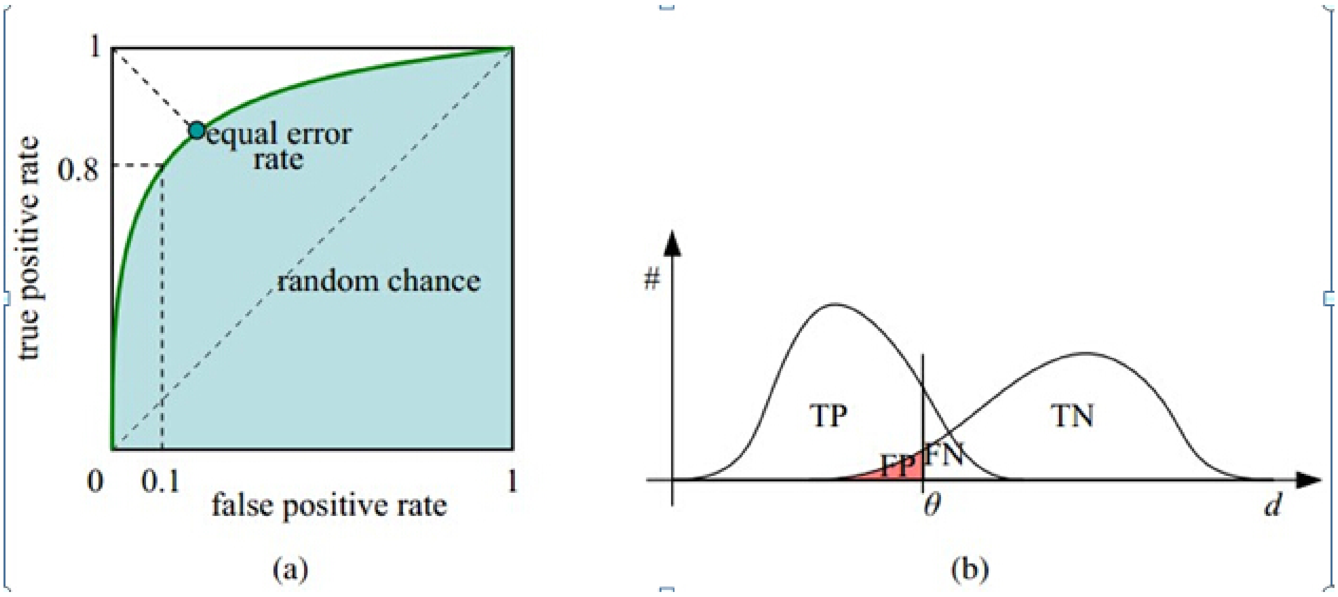
```
[2 3] [0 1]
[0 1] [2 3]
[3 1] [2 0]
[2 0] [3 1]
```

3- 模型对比方法(Methods for Model Comparison)

比较两个模型的性能

Receiver Operating Characteristic (ROC)

1. Developed in 1950s for signal detection theory to analyze noisy signals(Characterize the trade-off between positive hits and false alarms)
2. ROC curve plots TP (on the y-axis) against FP (on the x-axis)
3. Performance of each classifier represented as a point on the ROC curve.(Changing the threshold of algorithm, sample distribution or cost matrix changes the location of the point)



1. (0,0): declare everything to be negative class
2. (1,1): declare everything to be positive class
3. (0,1): ideal

AUG

Area Under the ROC (AUC) curve

1. Ideal: Area = 1
2. Random guess: Area = 0.5

Test of Significance

Given two models:

1. Model M1: accuracy = 85%, tested on 30 instances
2. Model M2: accuracy = 75%, tested on 5000 instances

Can we say M1 is better than M2?

1. How much confidence can we place on accuracy of M1 and M2?
2. Can the difference in performance measure be explained as a result of random fluctuations in the test set?

准确度的置信区间(Confidence Interval for Accuracy)

Prediction can be regarded as a Bernoulli trial

1. A Bernoulli trial has 2 possible outcomes
2. Possible outcomes for prediction: correct or wrong
3. Collection of Bernoulli trials has a Binomial distribution(二项分 布):

Given x (of correct predictions) or equivalently,$acc = x/N$, and N (of test instances), Can we predict p (true accuracy of model)?

For large test sets (N > 30) acc has a normal distribution with mean p and variance $p(1-p)/N$

$$P\left(Z_{\alpha/2} < \frac{acc - p}{\sqrt{p(1-p)/N}} < Z_{\alpha/2}\right) = 1 - \alpha$$

Confidence Interval for p:

$$P = \frac{2 * N * acc + Z_{\alpha/2}^2 \pm Z_{\alpha/2}\sqrt{Z_{\alpha/2}^2 + 4 * N * acc - 4 * N * acc^2}}{2(N + Z_{\alpha/2}^2)}$$

例子:

Consider a model that produces an accuracy of 80% when evaluated on 100 test instances:

1. N=100,acc=0.8
2. Let $1-\alpha$=0.95(95% confidence)
3. From probability table, $Z_{\alpha/2} = 1.96$

1-α	**Z**
0.99	2.58
0.98	2.33
0.95	1.96
0.90	1.65

可以计算$P_{lower} = 0.711$,$P_{upper} = 0.866$

Given two models, say M1 and M2, which is better?

1. M_1 is tested on D_1 (size = n_1), found error rate = e_1
2. M_2 is tested on D_2 (size = n_2), found error rate = e_2
3. Assume D_1 and D_2 are independent
4. If n_1 and n_2 are sufficiently large, then

$$e_1 \sim N(\mu_1, \sigma_1)$$

$$e_2 \sim N(\mu_2, \sigma_2)$$

5. Approximate:

$$\hat{\sigma_i} = \frac{e_i(1-e_i)}{n_i}$$

To test if performance difference is statistically significant: $d = e_1 - e_2$

1. $d \sim N(d_t, \sigma_t)$ where d_t is the true difference
2. Since D_1 and D_2 are independent, their variance adds up:

$$\sigma_t^2 = \sigma_1^2 + \sigma_2^2 \approx \hat{\sigma_1}^2 + \hat{\sigma_2}^2 = \frac{e_1(1-e_1)}{n_1} + \frac{e_2(1-e_2)}{n_2}$$

3. At $(1-\alpha)$ confidence level, the confidence interval for the true difference d_t is given by:

$$d_t = d \pm Z_{\alpha/2}^2 \hat{\sigma}_t$$

例: Given $M_1 : n_1 = 30$, $e_1 = 0.15$ M_2: $n_2 = 5000$, $e_2 = 0.25$

$d = |e_2 - e_1| = 0.1$(2-sided test)

$$\hat{\sigma_d} = \frac{0.15(1-0.15)}{30} + \frac{0.25(1-0.25)}{5000} = 0.0043$$

At 95% confidence level, $Z_{\alpha/2} = 1.96$

$$d_t = 0.100 \pm 1.96 * \sqrt{0.0043} = 0.100 \pm 0.128$$

Interval contains 0 $\Rightarrow$ difference may not be statistically significant