Vbubblery / review_IOT_Streaming

Branch: master review_IOT_Streaming / README.md Find file Copy path

Vbubblery Update README.md 9942e64 Feb 5, 2018

1 contributor

358 lines (314 sloc) 14.4 KB

<style> img{ width: 40%; } </style>

IoT Data Stream Mining

Data Streams

- 序列可能是无限的
- 高数据量：线性空间
- 高速到达：以次线性时间为例
- 一旦流的元素已经被处理，它将被丢弃或归档

MORRIS APPROXIMATE COUNTING ALGORITHM

```
Init counter c ← 0 //初始一个 计数器
  for every event in the stream //对于在流中收到的每一个event做如下操作
    do rand = random number between 0 and 1 //先随机一个数字
      if rand < p //如果这个数字小于某一个既定的概率值
      then c ← c + 1 //则让加速器+1
```

然后就是去除以这个概率获得一个相对近似的值 比如说1/2的话 就是 c/0.5

Find number of distinct items

- Flajolet-Martin算法 不重复的数字的个数
  - https://greatpowerlaw.wordpress.com/2012/10/14/flajoletmartin/
  - stream:{4,1,2,3,2,4}
  - binary:{100,001,010,011,100}
  - r(a) = {2,0,1,0,2}
  - R = max(r(a)) = 2
  - Estimate = 2^R = 2 ^ 2 = 4

```
  Init bitmap[0 . . . L − 1] ← 0    //初始化一个位图
  for every item x in the stream    // 对于在流中收到的每一个item
    do index = ρ(hash(x))           //首先转换成hash值的二进制，然后从左往右找出最近的一个1的位置
      if bitmap[index] = 0          // 如果位图上这个位置的值为0，则变为1
        then bitmap[index] = 1
  b ← position of leftmost zero in bitmap // 为图上从左往右 最近的一个0的位置
  return 2**b/0.77351 //计算获得不同的个数


Init M ← −∞             //初始化一个M为负无穷
  for every item x in the stream
    do M = max(M, ρ(h(x))    //M 为本身或者，hash值的二进制，然后从左往右找出最近的一个1的位置
  b ← M + 1 // 也就是最远的1的index+1
  return 2 ** b/0.77351
```

item x	$hash(x)$	$\rho(hash(x))$	bitmap
a	0110	1	01000
b	1001	0	11000
c	0111	1	11000
d	1100	0	11000
a			
b			
e	0101	1	11000
f	1010	0	11000
a			
b			

  - $$b = 2, n \approx 2^2/0.77351 = 5.17$$

- HYPERLOGLOG COUNTER
  - 演示地址：http://content.research.neustar.biz/blog/hll.html
  - 解释地址：https://www.youtube.com/watch?v=QSy5Y1lZtcs

HYPERLOGLOG COUNTER

```
1  Init M[0 ... b − 1] ← −∞
2  for every item x in the stream
3       do index = h_b(x)
4            M[index] = max(M[index], ρ(h^b(x)))
5  return α_m m^2 / Σ_{j=0}^{m−1} 2^{−M[j]}
```

  - $$h(x) = 010011000111$$
    $$h_3(x) = 010 \text{ and } h^3(x) = 011000111$$
  - HLL和LL的区别在于最后的return的公司不一样
  - 首先我们初始化一个size为b的list，这个b为我们设置的值（调参）
  - 然后 定位到010 (2)这个index, 然后去寻找011000111中左往右最近的一个1，这里为2
  - 看下M[index]中的值，如果比这个小，就更新为这个

Find most frequent items

- MAJORITY 最多出现的一个item

```
Init counter c ← 0    // 初始化一个int
  for every item s in the stream
    do if counter is zero //如果c为0
        then pick up the item //就选择这个item
    do if item is the same // 如果相同
        then increment counter //+1
        else decrement counter //-1
```

  - 最后出现的这额能是最多的，因为如果他出现的次数最多，他肯定可以覆盖别的
  - 1,2,2,3,3,3,1 =>
  - 1,0,1,0,1,2,0

- FREQUENT

```
for every item i in the stream
  do if item i is not monitored // top 10 监视区域
    do if < k items monitored //如果监视区域未满，且不在里面，则放进去并+1
      then add a new item with count 1
      else if an item z whose count is zero exists //如果满了，则看谁的count为0，如果为0则删除
        then replace this item z by the new one //并将其替换为新的一个item
        else decrement all counters by one //否则都+1
  else // 这个item已经在监视区域里面了
    increase its counter by one //全-1
```

- LOSSYCOUNTING这个不太懂

LOSSYCOUNTING

```
1  for every item i in the stream
2       do if item i is not monitored
3             then add a new item with count 1 + Δ
4             else ▷ item i is monitored
5                  increase its counter by one
6          if ⌊n/k⌋ ≠ Δ
7            then Δ = ⌊n/k⌋
8                  decrement all counters by one
9                  remove items with zero counts
```

- SPACE SAVING

```
for every item i in the stream
  do if item i is not monitored // top 10 监视区域
    do if < k items monitored //如果监视区域未满，且不在里面，则放进去并+1
      then add a new item with count 1
      else replace the item with lower counter //否则将最低的counter对应的item替换掉
        increase its counter by one //并counter++
    else ⋉ item i is monitored //如果他在监视区域里面
      increase its counter by one //就对其++
```

  - CM-Sketch
    - computes frequency data adding and removing real values
    - RESERVOIR SAMPLING

      a. 申请一个长度为k的数组reservoir保存抽样。

b. 保存首先接收到的k个元素

c. 当接收到第i个新元素t时，以k/i的概率随机替换reservoir中的元素(即生成[1,i]间随机数j，若j<=k，则以t替换reservoir[j])

```
for every item i in the first k items of the stream //将stream的前k个储存到池里
  do store item i in the reservoir
n = k //令N = K
for every item i in the stream after the first k items of the stream //K之后的
  do select a random number r between 1 and n //从1-n中随机数
    if r < k //如果随机数比设置的k要小
      then replace item r in the reservoir with item i //然后replace[r]=item_i
    n = n + 1 //n++
```

Mean and Variance 均值与方差

- Given a stream x1, x2, . . . , xn
- $$\bar{x}_n = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$$

$$\sigma_n^2 = \frac{1}{n-1} \cdot \sum_{i=1}^{n} (x_i - \bar{x}_i)^2.$$

- $$s_n = \sum_{i=1}^{n} x_i,\ q_n = \sum_{i=1}^{n} x_i^2$$

$$s_n = s_{n-1} + x_n,\ q_n = q_{n-1} + x_n^2$$

$$\bar{x}_n = s_n / n$$

$$\sigma_n^2 = \frac{1}{n-1} \cdot (\sum_{i=1}^{n} x_i^2 - n\bar{x}_i^2) = \frac{1}{n-1} \cdot (q_n - s_n^2/n)$$

- q是什么意思？？？？

Data Stream Sliding Window 数据流滑动窗口

- 10110001111 0101011 => 101100011110 1010111 => 1011000111101 0101110
- We can maintain simple statistics over sliding windows, using $O(\frac{1}{\epsilon} \log^2 N)$ space
  - N is the length of the sliding window
  - ε is the accuracy parameter
- Exponential Histograms
  - https://books.google.fr/books?id=uwiwHFLbbDAC&pg=PA62&lpg=PA62&dq=%3C+content+of+the+last+bucket+W/M&source=bl&ots=9RDl6yAirn&sig=YmNyb88-uG1ewUo6tHICCcS6fdc&hl=en&sa=X&ved=0ahUKEwiIx9i2qorZAhXkDMAKHRxuBdcQ6AEIKTAA#v=onepage&q&f=false

$M = 2$

1010101	101	11	1	1	1
Content: 4	2	2	1	1	1
Capacity: 7	3	2	1	1	1

1010101	101	11	11	1
Content: 4	2	2	2	1
Capacity: 7	3	2	2	1

1010101	10111	11	1
Content: 4	4	2	1
Capacity: 7	5	2	1

- In our example, suppose M = 2, if a new element "1" arrives then

1010101	101	11	1	1	1
Content: 4	2	2	1	1	1
Capacity: 7	3	2	1	1	1

There are 3 buckets of 1, so we compress it:

1010101	101	11	11	1
Content: 4	2	2	2	1
Capacity: 7	3	2	2	1

and now as we have 3 buckets of size 2, we compress it again

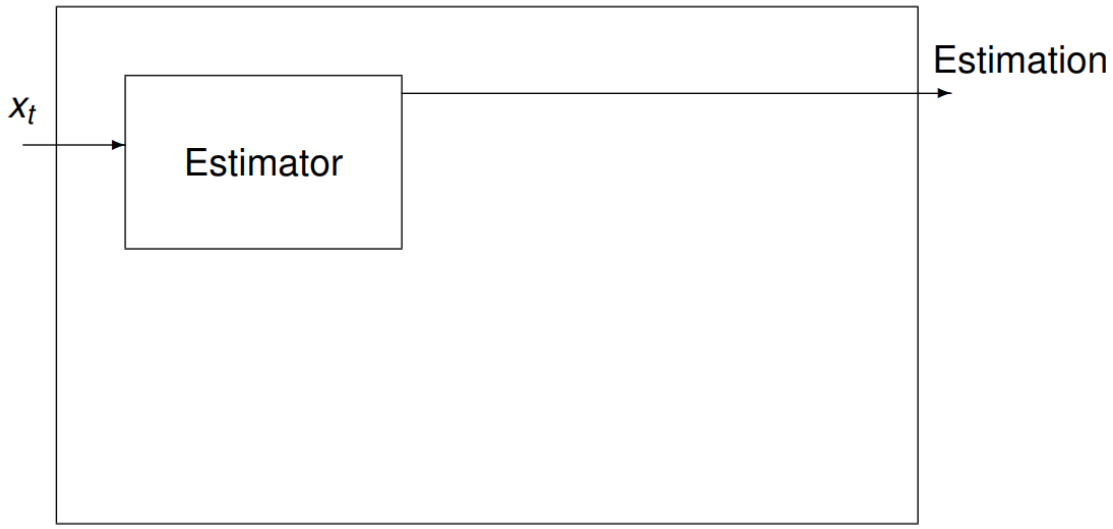
1010101	10111	11	1
Content: 4	4	2	1
Capacity: 7	5	2	1

And finally, if we detect change, we reduce the size of our sliding window deleting the last bucket:

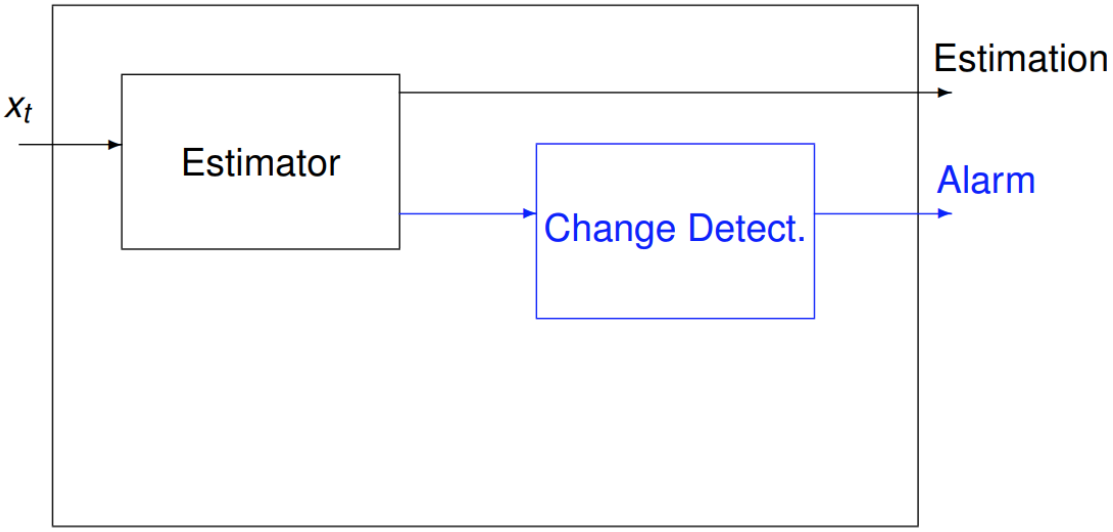
10111	11	1
Content: 4	2	1
Capacity: 5	2	1

Concept Drift

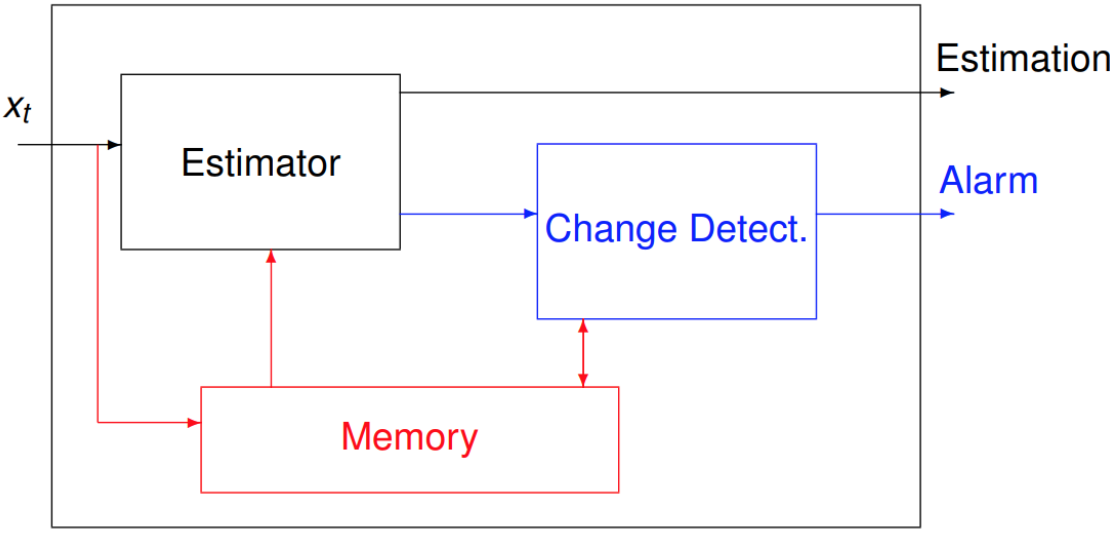
- 概念漂移意味着模型尝试预测的目标变量的统计特性随着时间的推移以不可预知的方式发生变化。 这会导致因为随着时间的推移，预测变得不准确的问题。
- 在商品销售应用中，概念漂移的一个原因可能是季节性，这意味着购物行为季节性变化。 例如，冬季假期的销售额可能比夏季高



- 普通预测器进行预测



- 检测到变化时预警



- 应该是去操作内存 内存交互?

Concept Drift Evaluation

- change detector 是用来检测真实的变化并且避免错误的预警的一种折中的设计所以我们需要满足几个条件:
  - 高准确率

- 低平均检测时间，
- 低假阳率（医学上：误判断其有病的概率 详细：http://blog.csdn.net/luo123n/article/details/48573397）
- 低missed检测率
- 低空间开销与时间花费
- 要有理论做支撑
- 不需要参数设置
- CUSUM

$$g_0 = 0, \qquad g_t = \max(0, g_{t-1} + \epsilon_t - \upsilon)$$

  - $$\text{if } g_t > h \text{ then alarm and } g_t = 0$$

    v和h是超参数
- Page Hinckley 这个不太明白

$$g_0 = 0, \qquad g_t = g_{t-1} + (\epsilon_t - \upsilon)$$

$$G_t = \min(g_t)$$

  - $$\text{if } g_t - G_t > h \text{ then alarm and } g_t = 0$$
- Geometric Moving Average

$$g_0 = 0, \qquad g_t = \lambda g_{t-1} + (1-\lambda)\epsilon_t$$

  - $$\text{if } g_t > h \text{ then alarm and } g_t = 0$$

    这里面的λ是遗忘变量，被用作给上一个数据多或少权值
- statistical test

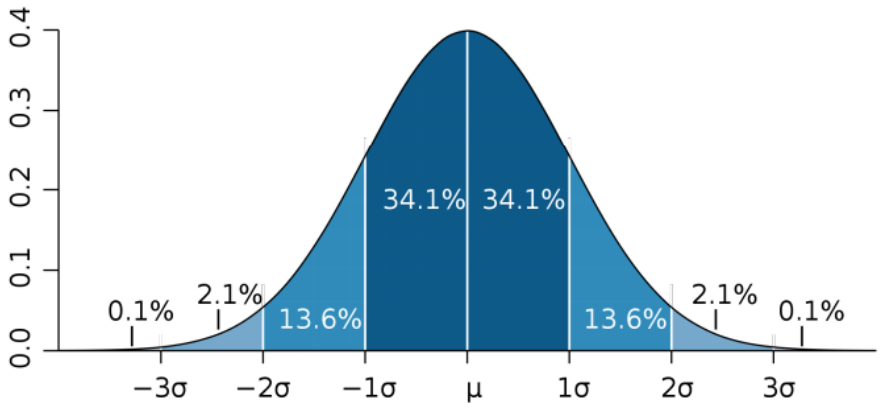
$$\hat{\mu}_0 - \hat{\mu}_1 \in N(0, \sigma_0^2 + \sigma_1^2), \text{ under } H_0$$

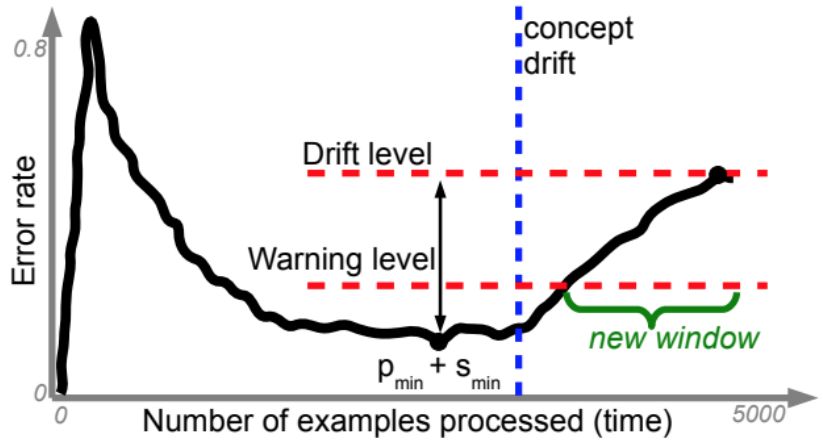
Example: Probability of false alarm of 5%

$$\Pr\left(\frac{|\hat{\mu}_0 - \hat{\mu}_1|}{\sqrt{\sigma_0^2 + \sigma_1^2}} > h\right) = 0.05$$

As $P(X < 1.96) = 0.975$ the test becomes

$$\frac{(\hat{\mu}_0 - \hat{\mu}_1)^2}{\sigma_0^2 + \sigma_1^2} > 1.96^2$$

- 

with 6 sigma
- 

- ADWIN: Adaptive Data Stream Sliding Window

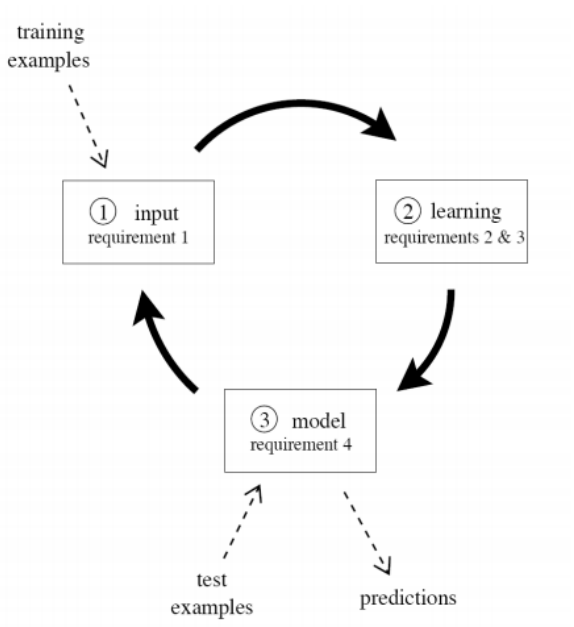
Let $W =$ 101010110111111

- Equal & fixed size subwindows: 1010 1011011 1111
- Equal size adjacent subwindows: 1010101 1011 1111
- Total window against subwindow: 10101011011 1111
- ADWIN: All adjacent subwindows:

1	01010110111111
1010	10110111111
1010101	10111111
1010101101	11111
10101011011111	1

- $O(\frac{1}{\epsilon}\log^2 N)$ space
  - 我们可以通过滑动窗口保持简单的统计,Using , N 是滑窗的长度 另一个是准确度参数
  - 后面是算法和一些理论，**没太明白，但是应该挺重要的！！！！！！！！！！！！！**

Evaluation



- i. 一次处理一个例子，只检查一次（最多）
  ii. 使用有限的内存
  iii. 在有限的时间中工作
  iv. 随时准备好预测

Evaluation Framework

1. Error estimation: Hold-out or Prequential 误差估计
2. Evaluation performance measures: Accuracy or κ-statistic 评估绩效评估
3. Statistical significance validation: MacNemar or Nemenyi test 统计显着性验证

Error Estimation

- Holdout Evaluation
  - 有可用于测试的数据
    a. 保留一个独立的测试集
    b. 将当前的决策模型定期应用于测试集
    c. 保留下来的估计损失是一个无偏估计量
  - 没有可用于测试的数据 Prequential or Interleaved-Test-Then-Train
    a. 从一系列例子中计算出一个模型的误差。
    b. 对于流中的每个item，实际模型都会进行预测，然后使用它来更新模型。
  - Hold-out is more accurate, but needs data for testing
    - 所以我们用prequential 去接近Hold-out
    - 使用滑动窗口或衰落因子估计精度

Evaluation performance measures

	Predicted Class+	Predicted Class-	Total
Correct Class+	75	8	83
Correct Class-	7	10	17
Total	82	18	100

- Accuracy = $\frac{75}{100} + \frac{10}{100} = \frac{75}{83}\frac{83}{100} + \frac{10}{17}\frac{17}{100} = 85\%$
- Arithmetic mean = $(\frac{75}{83} + \frac{10}{17})/2 = 74.59\%$
- Geometric mean = $\sqrt{\frac{75}{83}\frac{10}{17}} = 72.90\%$

Performance Measures with Unbalanced Classes

- Kappa
  - $\kappa = \frac{p_0 - p_c}{1 - p_c}$
  - p0 classifier's prequential accuracy (75+10)/100=0.85
  - pc 分类器做出正确预测的概率 (82*83+18*17)/100/100=0.7112
  - k = 0.85-0.7112/1-0.7112=48%
- McNemar test

	Classifier A Class+	Classifier A Class-	Total
Classifier B Class+	c	a	c+a
Classifier B Class-	b	d	b+d
Total	c+b	a+d	a+b+c+d

  - M = |a – b – 1| ** 2 /(a + b)
  - 在X^2分布下(卡布分布) confidence 置信度在0.99的时候，如果M> 6.635，则拒绝零假设（性能相等）
- Nemenyi test
  - 没太明白
- Cost Evaluation Example

	Accuracy	Time	Memory
Classifier A	70%	100	20
Classifier B	80%	20	40

  - Which classifier is performing better?

	Accuracy	Time	Memory	**RAM-Hours**
Classifier A	70%	100	20	**2,000**
Classifier B	80%	20	40	**800**

Which classifier is performing better?

Ensemble Methods

Bagging

- 现在我有4个数据：A,B,C,D
- Classifier 1 使用的是B,A,C,B => A(1) B(2) C(1) D(0)
- Classifier 2 使用的是D,B,A,D => A(1) B(1) C(0) D(2)
- Classifier 3: B, A, C, B
- Classifier 4: B, C, B, B
- Classifier 5: D, C, A, C
- Bagging 就是构建一些列基本models，用通过替换抽取随机样本的方式创建boostrap样本
- 每个基本模型的训练集包含每个原始训练样例K次，其中P（K = k）遵循二项分布。(二项分布就是重复n次独立的伯努利试验。在每次试验中只有两种可能的结果，而且两种结果发生与否互相对立，并且相互独立，与其它各次试验结果无关，事件发生与否的概率在每一次独立试验中都保持不变) 当二项分布的n很大而p很小时，泊松分布可作为二项分布的近似，其中λ为np。通常当n≧20,p≦0.05时，就可以用泊松公式近似得计算
- Oza and Russell

1: Initialize base models h_m for all $m \in \{1, 2, ..., M\}$
2: **for all** training examples **do**
3: **for** $m = 1, 2, ..., M$ **do**
4: Set $w =$ *Poisson*(1)
5: Update h_m with the current example with weight w
6: **anytime output:**
7: **return** hypothesis: $h_{fin}(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} I(h_t(x) = y)$

- Hoeffding Option Tree
  - 包含附加选项节点的常规Hoeffding树允许应用多个测试，从而将多个Hoeffding树作为单独的路径。
- Random Forests
  i. 输入训练集通过替换采样获得 (bagging)
  ii. 树的节点只能使用固定数量的随机属性进行拆分
  iii. 没有进行剪枝

Accuracy Weighted Ensemble (后期补充解释，并给出例子)

- Process chunks of instances of size W
- Builds a new classifier for each chunk
- Removes old classifier
- Weight each classifier using error

$$w_i = MSE_r - MSE_i$$

where

$$MSE_r = \sum_c p(c)(1 - p(c))^2$$

and

$$MSE_i = \frac{1}{|S_n|} \sum_{(x,c) \in S_n} (1 - f_c^i(x))^2$$

ADWIN Bagging (后期补充解释，并给出例子)

- 懂了adwin再说

Leveraging Bagging for Evolving Data Streams 利用bagging进化数据流 (后期补充解释，并给出例子)

- Leveraging Bagging
  - Using Poisson(λ) 泊松分布的参数λ是单位时间(或单位面积)内随机事件的平均发生率

    观察事物平均发生m次的条件下，实际发生x次的概率P（x）可用下式表示：

    $$P(x) = \frac{m^x}{x!} \times e^{-m}$$

  - $$p(0) = e^{-m}$$
- Leveraging Bagging MC
  - Using Poisson(λ) and Random Output Codes
- Fast Leveraging Bagging ME
  - if an instance is misclassified: weight = 1
  - if not: weight = eT /(1 – eT),
- Empirical evaluation 实证评估

	Accuracy	RAM-Hours
Hoeffding Tree	74.03%	0.01
Online Bagging	77.15%	2.98
ADWIN Bagging	79.24%	1.48
Leveraging Bagging	85.54%	20.17
Leveraging Bagging MC	85.37%	22.04
Leveraging Bagging ME	80.77%	0.87

Boosting (后期补充解释，并给出例子)

- Boosting算法将弱model转化为强大的model

Oza and Russell's *Online Boosting*

Initialize base models h_m for all $m \in \{1, 2, ..., M\}$, $\lambda_m^{sc} = 0, \lambda_m^{sw} = 0$
for all training examples **do**
  Set "weight" of example $\lambda_d = 1$
  for $m = 1, 2, ..., M$ **do**
    Set $k = Poisson(\lambda_d)$
    for $n = 1, 2, ..., k$ **do**
      Update h_m with the current example
    if h_m correctly classifies the example **then**
      $\lambda_m^{sc} \leftarrow \lambda_m^{sc} + \lambda_d$
      $\epsilon_m = \frac{\lambda_m^{sw}}{\lambda_m^{sw} + \lambda_m^{sc}}$
      $\lambda_d \leftarrow \lambda_d \left(\frac{1}{2(1-\epsilon_m)}\right)$ Decrease λ_d
    else
      $\lambda_m^{sw} \leftarrow \lambda_m^{sw} + \lambda_d$
      $\epsilon_m = \frac{\lambda_m^{sw}}{\lambda_m^{sw} + \lambda_m^{sc}}$
      $\lambda_d \leftarrow \lambda_d \left(\frac{1}{2\epsilon_m}\right)$ Increase λ_d
anytime output:
return hypothesis: $h_{fin}(x) = \arg\max_{y \in Y} \sum_{m:h_m(x)=y} -\log \epsilon_m/(1-\epsilon_m)$

Stacking

- 使用分类器来组合基本分类器的预测
- Restricted Hoeffding Trees

Trees for all possible attribute subsets of size k

- $\binom{m}{k}$ subsets
- $\binom{m}{k} = \frac{m!}{k!(m-k)!} = \binom{m}{m-k}$

Example for 10 attributes

$$\binom{10}{1} = 10 \quad \binom{10}{2} = 45 \quad \binom{10}{3} = 120$$

$$\binom{10}{4} = 210 \quad \binom{10}{5} = 252$$

  - 树的大小为k,总属性为m的所有可能的属性子集

Classification

- multi-label Classification 每个实例多个标签而不是一个标签 y {sunset, people, foliage, beach, urban, field} {0，1}**6 = [1,0,1,0,0,0]
- Single-label vs. Multi-label

	$K = 2$	$K > 2$
$L = 1$ (single-label)	**binary**	**multi-class**
$L > 1$ (multi-label)	**multi-label**	**multi-output**†

† aka multi-objective, multi-target, multi-dimensional, multi-label.

  - Figure: For L labels (target variables), each of K values.

Table: Single-label $Y \in \{0, 1\}$

X_1	X_2	X_3	X_4	X_5	Y
1	0.1	3	1	0	0
0	0.9	1	0	1	1
0	0.0	1	1	0	0
1	0.8	2	0	1	1
1	0.0	2	0	1	0
0	0.0	3	1	1	?

Table: Multi-label $[Y_1, \ldots, Y_L] \in 2^L$

X_1	X_2	X_3	X_4	X_5	Y_1	Y_2	Y_3	Y_4
1	0.1	3	1	0	0	1	1	0
0	0.9	1	0	1	1	0	0	0
0	0.0	1	1	0	0	1	0	0
1	0.8	2	0	1	1	0	0	1
1	0.0	2	0	1	0	0	0	1
0	0.0	3	1	1	?	?	?	?

- Notation
  - L = {sunset, people, foliage, beach, urban, field}
  - xi = 
  - produce predictions:
    - [1, 0, 1, 0, 0, 0] ⇔ {sunset, foliage}

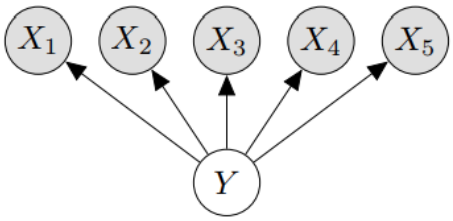
Text Categorization and Tag Recommendation

- 比如说在IMDb数据集中，同一个样本的label 可能既是恐怖片也是动作片，如下：

i	*abandoned* X_1	*accident* X_2	$\ldots$	*violent* X_{1000}	*wedding* X_{1001}	horror Y_1	romance Y_2	$\ldots$	comedy Y_{27}	action Y_{28}
1	1	0	$\ldots$	0	1	0	1	$\ldots$	0	0
2	0	1	$\ldots$	1	0	1	0	$\ldots$	0	0
3	0	0	$\ldots$	0	1	0	1	$\ldots$	0	0
4	1	1	$\ldots$	0	1	1	0	$\ldots$	0	1
5	1	1	$\ldots$	0	1	0	1	$\ldots$	0	1
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
120919	1	1	$\ldots$	0	0	0	0	$\ldots$	0	1

Probabilistic Models

- (Single-label) Naive Bayes



- P(Y|X)是指在已知X的情况下，Y的概率是多少
- P(X) 是指在所有的数据集中，当前样本的概率
- P(Y) 是指label中的选项在所有数据集中的比例
- P(X|Y) 是指在已知某一label的情况下，得出的样本是正确样本的概率
  - 然后我们假设变量X间是条件独立的

$$P(\mathbf{X}|Y) = \prod_{d=1}^{D} P(X_d|Y)$$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

最后以最大值为结果

- Logistic Regression逻辑回归
  - 自己看公式

Clustering

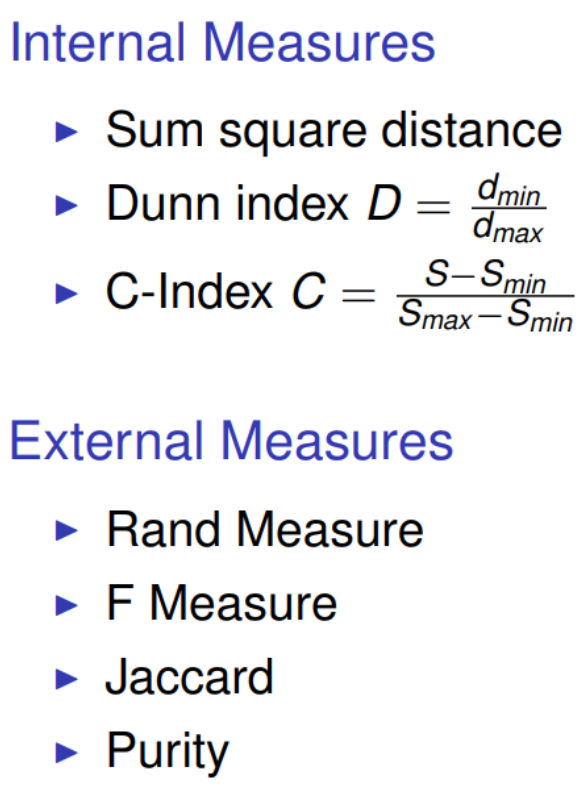
k-means

- Choose k initial centers C = {c1, . . . , ck }
- 当停止条件没有被满足时
  - I For i = 1, . . . , N
    - find closest center ck ∈ C to each instance pi
    - assign instance pi to cluster Ck
  - For k = 1, . . . , K
    - set ck to be the center of mass of all points in C

k-means++

- Choose a initial center c1
- For k = 2, . . . , K
  - select ck = p ∈ I with probability d**2(p, C)/cost(C, I)
- 剩下的和k-means一样

性能测试：



-

BIRCH 之前讲过

Clu-Stream

- 使用微型群集在线存储统计信息
  - Clustering Features CF = (N, LS, SS, LT, ST)
  - N: numer of data points 数据点的数量
  - LS: linear sum of the N data points N个数据点的线性和
  - SS: square sum of the N data points N个数据点的平方和
  - LT: linear sum of the time stamps 时间戳的线性总和
  - ST: square sum of the time stamps 时间戳的平方总和
- 在线阶段
  - 对于到达的每一个数据点
    - 这一点被一个micro-cluster吸收
    - 这一点就开始了自己的一个新的micro-cluster集群
      - delete oldest micro-cluster
      - merge two of the oldest micro-cluster
- 离线阶段
  - Apply k-means using micro-clusters as points

Density based methods

- 基本的讲过了
- DenStream
  - A-邻域（p）：距p小于或等于A的点集合
  - 核心对象：其A领域的整体权重至少为μ的对象
  - 密度区域：核心对象A邻域的联合

For a group of points $p_{i_1}, p_{i_2}, \ldots, p_{i_n}$,
with time stamps $T_{i_1}, T_{i_2}, \ldots, T_{i_n}$

- core-micro-cluster
  - $w = \sum_{j=1}^{n} f(t - T_{i_j})$ where $f(t) = 2^{-\lambda t}$ and $w \geq \mu$
  - $c = \sum_{j=1}^{n} f(t - T_{i_j}) p_{i_j} / w$
  - $r = \sum_{j=1}^{n} f(t - T_{i_j}) dist(p_{i_j}, c) / w$ where $r \leq \epsilon$
- potential core-micro-cluster
  - $w = \sum_{j=1}^{n} f(t - T_{i_j})$ where $f(t) = 2^{-\lambda t}$ and $w \geq \beta\mu$
  - $\overline{CF^1} = \sum_{j=1}^{n} f(t - T_{i_j}) p_{i_j}$
  - $\overline{CF^2} = \sum_{j=1}^{n} f(t - T_{i_j}) p_{i_j}^2$ where $r \leq \epsilon$
- outlier micro-cluster: $w < \beta\mu$

- 在线阶段：
  - For each new point that arrives
    - try to merge to a p-micro-cluster
    - else, try to merge to nearest o-micro-cluster
      - if $w > \beta\mu$ then
      - convert the o-micro-cluster to p-micro-cluster
    - otherwise create a new o-microcluster
- 离线阶段：
  - for each p-micro-cluster c_p
    - if $w < \beta\mu$ then remove c_p
  - for each o-micro-cluster c_o
    - if $w < (2^{-\lambda(t - t_o + T_p)} - 1)/(2^{-\lambda T_p} - 1)$ then remove c_o
  - Apply DBSCAN using microclusters as points

ClusTree

StreamKM++