Ensemble Methods

Albert Bifet (@abifet)

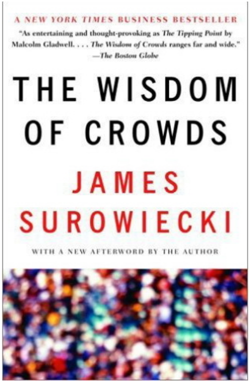
Paris, 6 December 2016

albert.bifet@telecom-paristech.fr

Data Streams

Big Data & Real Time

Ensemble Learning: The Wisdom of Crowds


A NEW YORK TIMES BUSINESS BESTSELLER
"As entertaining and thought-provoking as The Tipping Point by Malcolm Gladwell. . . . The Wisdom of Crowds ranges far and wide."
—The Boston Globe
THE WISDOM OF CROWDS
JAMES SUROWIECKI
WITH A NEW AFTERWORD BY THE AUTHOR

Diversity of opinion, Independence
Decentralization, Aggregation

Bagging

Example

Dataset of 4 Instances : A, B, C, D

Classifier 1: B, A, C, B
Classifier 2: D, B, A, D
Classifier 3: B, A, C, B
Classifier 4: B, C, B, B
Classifier 5: D, C, A, C

Bagging builds a set of *M* base models, with a bootstrap sample created by drawing random samples with replacement.

Bagging

Example

Dataset of 4 Instances : A, B, C, D

Classifier 1: A, B, B, C
Classifier 2: A, B, D, D
Classifier 3: A, B, B, C
Classifier 4: B, B, B, C
Classifier 5: A, C, C, D

Bagging builds a set of M base models, with a bootstrap sample created by drawing random samples with replacement.

Bagging

Example

Dataset of 4 Instances : A, B, C, D

Classifier 1: A, B, B, C: A(1) B(2) C(1) D(0)
Classifier 2: A, B, D, D: A(1) B(1) C(0) D(2)
Classifier 3: A, B, B, C: A(1) B(2) C(1) D(0)
Classifier 4: B, B, B, C: A(0) B(3) C(1) D(0)
Classifier 5: A, C, C, D: A(1) B(0) C(2) D(1)

Each base model's training set contains each of the original training example K times where $P(K = k)$ follows a binomial distribution.

Bagging

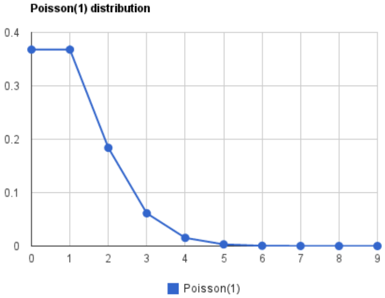


Figure: Poisson(1) Distribution.

Each base model's training set contains each of the original training example K times where $P(K = k)$ follows a binomial distribution.

Oza and Russell's *Online Bagging* for *M* models

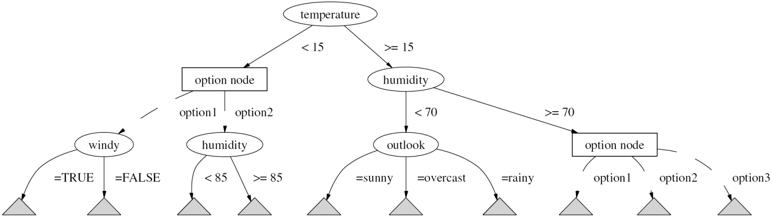
```
Initialize base models $h_m$ for all $m \in \{1, 2, ..., M\}$
for all training examples do
    for $m = 1, 2, ..., M$ do
        Set $w = Poisson(1)$
        Update $h_m$ with the current example with weight $w$

anytime output:
return hypothesis: $h_{fin}(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} I(h_t(x) = y)$
```

Hoeffding Option Tree

Hoeffding Option Trees

Regular Hoeffding tree containing additional option nodes that allow several tests to be applied, leading to multiple Hoeffding trees as separate paths.



Random Forests (Breiman, 2001)

Adding randomization to decision trees

- the input training set is obtained by sampling with replacement, like Bagging
- the nodes of the tree only may use a fixed number of random attributes to split
- the trees are grown without pruning

Accuracy Weighted Ensemble

Mining concept-drifting data streams using ensemble classifiers. Wang et al. 2003

- 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

ADWIN

An adaptive sliding window whose size is recomputed online according to the rate of change observed.

ADWIN has rigorous guarantees (theorems)

- On ratio of false positives and negatives
- On the relation of the size of the current window and change rates

ADWIN Bagging

When a change is detected, the worst classifier is removed and a new classifier is added.

ADWIN *Bagging* for *M* models

```
Initialize base models $h_m$ for all $m \in \{1, 2, ..., M\}$
for all training examples do
    for $m = 1, 2, ..., M$ do
        Set $w = Poisson(1)$
        Update $h_m$ with the current example with weight $w$
        if ADWIN detects change in error of one of the classifiers then
            Replace classifier with higher error with a new one

anytime output:
return hypothesis: $h_{fin}(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} I(h_t(x) = y)$
```

Leveraging Bagging for Evolving Data Streams

Randomization as a powerful tool to increase accuracy and diversity

There are three ways of using randomization:

- Manipulating the input data
- Manipulating the classifier algorithms
- Manipulating the output targets

Input Randomization

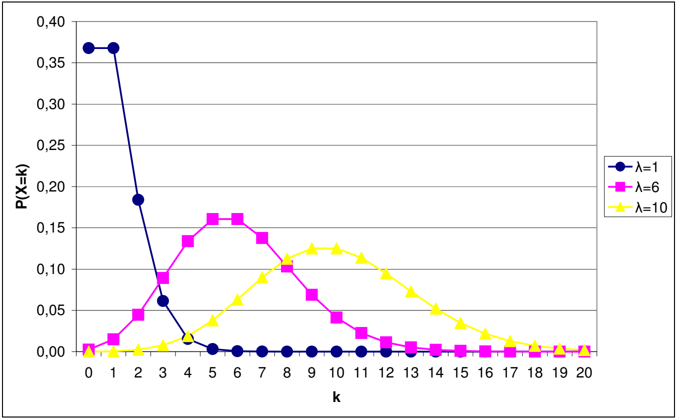


Figure: Poisson Distribution.

ECOC Output Randomization

Table: Example matrix of random output codes for 3 classes and 6 classifiers

	Class 1	Class 2	Class 3
Classifier 1	0	0	1
Classifier 2	0	1	1
Classifier 3	1	0	0
Classifier 4	1	1	0
Classifier 5	1	0	1
Classifier 6	0	1	0

Leveraging Bagging for Evolving Data Streams

Leveraging Bagging

- Using *Poisson*(λ)

Leveraging Bagging MC

- Using *Poisson*(λ) and Random Output Codes

Fast Leveraging Bagging ME

- if an instance is misclassified: weight = 1
- if not: weight = $e_T/(1 - e_T)$,

Empirical evaluation

	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

Leveraging Bagging

- Leveraging Bagging
  - Using $Poisson(\lambda)$
- Leveraging Bagging MC
  - Using $Poisson(\lambda)$ and Random Output Codes
- Leveraging Bagging ME
  - Using weight 1 if misclassified, otherwise $e_T/(1-e_T)$

Boosting

The strength of Weak Learnability, Schapire 90

A boosting algorithm transforms a weak learner into a strong one

Boosting

A formal description of Boosting (Schapire)

- given a training set $(x_1, y_1), \ldots, (x_m, y_m)$
- $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$
- for $t = 1, \ldots, T$
  - construct distribution D_t
  - find weak classifier

$$h_t : X \implies \{-1, +1\}$$

  with small error $\epsilon_t = \Pr_{D_t}[h_t(x_i) \neq y_i]$ on D_t
- output final classifier

Boosting

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 $\lambda_d$
        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 $\lambda_d$
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

Use a classifier to combine predictions of base classifiers

- Example: use a perceptron to do stacking

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

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

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