Multi-Label Classification with Boosting on Apache Spark

白刚 (@BaiGang-)

Sina Ad-Algo

March 21, 2015.

关于这一演示文稿

“抛砖引玉”，期待后续的讨论。
项目在https://github.com/baigang/spark_multiboost，欢迎大家参与。

LaTeX Beamer presentation.
https://github.com/BaiGang/slides/tree/master/spark_multiboost

About me:
BUAA → Yahoo! → OneBox search (360 so.com) → Sina/Weibo.

Overview

1. 背景
   - The problem
   - User Profiling
2. 问题与求解
   - 问题抽象
   - Preliminary: Boosting
   - Multi-label boosting
3. Multiboost on Spark
   - Strong learner on Apache Spark
   - Base learner on Apache Spark
   - Base learner: generalized binary classifier with vote vector
4. 后续工作

背景：The problem

Business

广告为媒体带来收益，同时也潜在的破坏用户体验。

背景：The problem

Business

广告为媒体带来收益，同时也潜在的破坏用户体验。

背景： The problem

Business

广告为媒体带来收益，同时也潜在的破坏用户体验。

不同的广告有不同的受众。
如何将广告投放给其对应的受众？

背景：User profiling

理解用户兴趣

一个用户属于哪个人群，是哪些广告的潜在受众。

背景: User profiling

Example

用户兴趣

背景: User profiling

Example

用户兴趣

User1

- 财经 -基金
- 财经 -股票
- 房产 -装修
- ...

背景: User profiling

Example

用户兴趣

User1

- 财经 -基金
- 财经 -股票
- 房产 -装修
- ...

User2

- 休闲 -境外游
- 娱乐 -综艺
- 休闲 -摄影
- ...

背景: User profiling

Example

用户兴趣

User1

- 财经 -基金
- 财经 -股票
- 房产 -装修
- ...

User2

- 休闲 -境外游
- 娱乐 -综艺
- 休闲 -摄影
- ...

User3

- 体育 -足球
- 体育 -NBA
- 游戏动漫
- ...

背景: User profiling

Example

用户兴趣

User1

- 财经 -基金
- 财经 -股票
- 房产 -装修
- ...

User2

- 休闲 -境外游
- 娱乐 -综艺
- 休闲 -摄影
- ...

User3

- 体育 -足球
- 体育 -NBA
- 游戏动漫
- ...

用户兴趣是多维度的 $\Rightarrow$ 标签集合

标签是根据 business 预先设定的

背景: User profiling

Example

用户兴趣

User1

- 财经 -基金
- 财经 -股票
- 房产 -装修
- …

User2

- 休闲 -境外游
- 娱乐 -综艺
- 休闲 -摄影
- …

User3

- 体育 -足球
- 体育 -NBA
- 游戏动漫
- …

用户兴趣是多维度的 $\Rightarrow$ 标签集合
标签是根据 business 预先设定的

我们要解决的实际问题: 如何给用户加对应的标签?

Overview

1. 背景
   - The problem
   - User Profiling
2. 问题与求解
   - 问题抽象
   - Preliminary: Boosting
   - Multi-label boosting
3. Multiboost on Spark
   - Strong learner on Apache Spark
   - Base learner on Apache Spark
   - Base learner: generalized binary classifier with vote vector
4. 后续工作

问题抽象

Model and prediction

根据给出的 user feature $\mathbf{x}$，输出符合其兴趣的标签集合 L.

$$\mathcal{F} : \mathcal{X} \to \mathcal{L}$$

问题抽象

Model and prediction

根据给出的 user feature $\mathbf{x}$，输出符合其兴趣的标签集合 L.

$$\mathcal{F} : \mathcal{X} \rightarrow \mathcal{L}$$

Model training

To infer a **vector-valued** function $\mathcal{F} : \mathcal{X} \rightarrow \mathcal{L}$ from a data set

$$\mathcal{D} = \{(\mathbf{x}_1, \mathbf{l}_1), \cdots, (\mathbf{x}_m, \mathbf{l}_m)\} \in (\mathcal{X} \times \mathcal{L})$$

, where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{l} \in \{+1, -1\}^L$, by minimizing **Hamming loss**:

$$\mathcal{Z}_H = \frac{1}{\|\mathcal{D}\|} \frac{1}{\|\mathcal{L}\|} \sum_{i=1}^{\|\mathcal{D}\|} \sum_{l=1}^{\|\mathcal{L}\|} \mathcal{I}[F(\mathbf{x}_i)_l \neq \mathbf{y}_{i,l}]$$

Multi-Label classification: Per-label bin-classification

为了得到这个 vector-valued function $\mathcal{F} : \mathcal{X} \to \mathcal{L}$, 我们为每个 $l \in \mathcal{L}$ 都训练一个 *binary* classifier，预测时将判断每一个标签的结果。

* One-versus-all implemented in LibSVM, scikit-learn, etc.

* Ad targeting 往往使用 per-campaign model，为每一个 ad compaign 训练一个二分类模型。

Multi-Label classification: Per-label bin-classification

为了得到这个 vector-valued function $\mathcal{F}:\mathcal{X}\to\mathcal{L}$, 我们为每个 $l\in\mathcal{L}$ 都训练一个 *binary* classifier，预测时将判断每一个标签的结果。

* One-versus-all implemented in LibSVM, scikit-learn, etc.

* Ad targeting 往往使用 per-campaign model，为每一个 ad compaign 训练一个二分类模型。

优缺点:

- 使用已有技术 ✓
  - LR, SVM 等二分类模型
- 易于验证 ✓
- Not (*economically*) scalable ×
  - num of labels: 10s $\to$ 10000s
  - 逐个训练低效、时间长

Multi-label classification: Scalable 方案

目标:

- 模型本身的输出就是多标签结果;
- 训练过程是最小化 Hamming loss;
- Scalable.

Multi-label classification: Scalable 方案

目标:

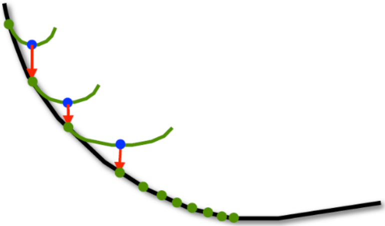
- 模型本身的输出就是多标签结果;
- 训练过程是最小化 Hamming loss;
- Scalable.

方案:

- "Improved Boosting Algorithm Using Confidence-Rated Predictions", Schapire & Singer, 1999.
  - 提出了 AdaBoost.MH 算法.
- "The return of AdaBoost.MH: multi-class Hamming trees", Kegl, 2014.
  - Factorization of base learners in AdaBoost.MH.
  - Decision stump & Hamming tree 作为 base learner.
- MultiBoost, http://multiboost.org
  - Open-sourced, single-machine implementations in CPP.

Preliminary: Boosting

- An additive model. $H(\mathbf{x}) = \sum_i^T \alpha_i h_i(\mathbf{x})$
- 迭代的在一个 (*re*-)weighted sample set 上去训练，
- 通过 reweighting，每次训练一个新模型去重点 fix 前一个模型分类错了的样本。



Preliminary: Adaptive Boosting

input : $\mathcal{D} = (x_i, y_i) \in \{(\mathcal{R}^n, \{-1, +1\})\},\ T,\ \mathfrak{B}$
output: $\mathbf{H} = \Sigma_{t=1}^{T} \alpha_t \mathcal{B}_t$
begin
    $\mathbf{W}_1(i) = 1/m$;
    for $t \leftarrow 1, \cdots, T$ **do**
        $\mathcal{B}_t \leftarrow \mathfrak{B}(\mathcal{D}, \mathbf{W}_t)$;
        Get hypothesis set: $h_t \leftarrow \mathcal{B}(\mathcal{D})$;
        Get the base coefficient α_t;
        Update the weights:

$$\mathbf{W}_{t+1}(i) = \mathbf{W}_t(i) \exp(-\alpha_t y_i h_t(x_i))/Z_t$$

        where Z_t is a normalization factor;
    end
end

AdaBoost.MH

MH: **M**ulti-class with **H**amming loss.

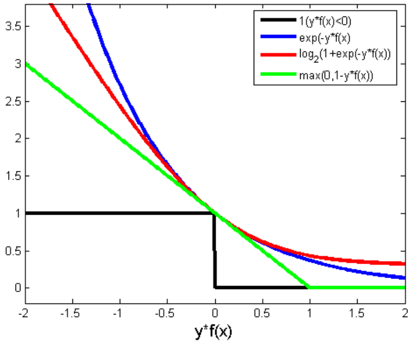
To produce a **vector-valued** discriminant function $\mathcal{F} : \mathcal{R}^n \to \mathcal{R}^K$ with a small **Hamming loss** $\hat{R}_H(\mathcal{F}, \mathbf{W})$ by minimizing the **weighted multi-class exponential margin-based error**

$$\hat{R}_{EXP}(\mathcal{F}, \mathbf{W}) = \frac{1}{mK} \Sigma_{i=1}^{m} \Sigma_{l=1}^{K} \mathbf{W}_{i,l} \exp(-\mathcal{F}_l y_{i,l})$$

The weights $\mathbf{W}$:

$$\Sigma_{i,l} \mathbf{W}_{i,l} = 1$$

Exponential loss vs Hamming loss



- Exponential loss "upper-bounds" Hamming loss.
  - Minimizing exp loss is minimizing Hamming loss.
- Exponential loss is convex.
  - Easy to optimize.

AdaBoost.MH

input : $\mathcal{D} = (x_i, y_i) \in \{(\mathcal{R}^n, \{-1, +1\}^K\}$, T, $\mathfrak{B}$
output: $\mathbf{H} = \Sigma_{t=1}^{T} \alpha_t \mathbf{B}_t$
begin
  $\mathbf{W}_1(i, l) = \frac{1}{mK}$;
  for $t \leftarrow 1, \cdots, T$ **do**
    Base learner and the edge: $(\mathcal{B}_t, \gamma) \leftarrow \mathfrak{B}(\mathcal{D}, \mathbf{W}_t)$;
    Get hypothesis set: $h_t \leftarrow \mathbf{B}(\mathcal{D})$;
    Get the base coefficient $\alpha_t \leftarrow \frac{1}{2} \log \frac{1+\gamma}{1-\gamma}$;
    Update the weights:

$$\mathbf{W}_{t+1}(i, l) = \mathbf{W}_t(i, l) \exp(-\alpha_t y_{i,l} h_t(x_i))/Z_t$$

    where Z_t is a normalization factor;
  end
end

AdaBoost.MH

input : $\mathcal{D} = (x_i, y_i) \in \{(\mathcal{R}^n, \{-1, +1\}^K\}$, T, $\mathfrak{B}$
output: $\mathbf{H} = \Sigma_{t=1}^{T} \alpha_t \mathbf{B}_t$
begin
  $\mathbf{W}_1(i, l) = \frac{1}{mK}$;
  for $t \leftarrow 1, \cdots, T$ **do**
    Base learner and the edge: $(\mathcal{B}_t, \gamma) \leftarrow \mathfrak{B}(\mathcal{D}, \mathbf{W}_t)$;
    Get hypothesis set: $h_t \leftarrow \mathbf{B}(\mathcal{D})$;
    Get the base coefficient $\alpha_t \leftarrow \frac{1}{2} \log \frac{1+\gamma}{1-\gamma}$;
    Update the weights:

$$\mathbf{W}_{t+1}(i, l) = \mathbf{W}_t(i, l) \exp(-\alpha_t y_{i,l} h_t(x_i))/Z_t$$

    where Z_t is a normalization factor;
  end
end

The base learner factor α

$$\begin{array}{c} \hat{R}_{EXP}(\alpha, \mathbf{h}, \mathbf{W}) = \sum_{i=1}^{n} \sum_{l=1}^{K} w_{i,l} \exp(-\alpha \mathbf{h}_{i,l} y_{i,l}) \\ = \sum_{i=1}^{n} \sum_{l=1}^{K} w_{i,l} (\mathcal{I}_{+} e^{-\alpha} + \mathcal{I}_{-} e^{\alpha}) \end{array},$$

with $\mathcal{I}_{+} = \mathcal{I}[y_{i,l} \mathbf{h}_{i,l} > 0], \mathcal{I}_{-} = \mathcal{I}[y_{i,l} \mathbf{h}_{i,l} < 0]$.

The base learner factor α

$$\begin{aligned}\hat{R}_{EXP}(\alpha, \mathbf{h}, \mathbf{W}) &= \sum_{i=1}^{n} \sum_{l=1}^{K} w_{i,l} \exp(-\alpha \mathbf{h}_{i,l} y_{i,l}) \\ &= \sum_{i=1}^{n} \sum_{l=1}^{K} w_{i,l} (\mathcal{I}_+ e^{-\alpha} + \mathcal{I}_- e^{\alpha})\end{aligned},$$

$$\text{with } \mathcal{I}_+ = \mathcal{I}[y_{i,l}\mathbf{h}_{i,l} > 0], \mathcal{I}_- = \mathcal{I}[y_{i,l}\mathbf{h}_{i,l} < 0].$$

Re-phrasing the EXP loss:

$$\begin{aligned}\hat{R}_{EXP}(\alpha, \mathbf{h}, \mathbf{W}) = \sum_{i=1}^{n} \sum_{l=1}^{K} w_{i,l} [\\ \frac{1}{2}(\mathcal{I}_+ e^{-\alpha} + \mathcal{I}_- e^{\alpha} + \mathcal{I}_+ e^{\alpha} + \mathcal{I}_- e^{-\alpha}) \\ +\frac{1}{2}(\mathcal{I}_+ e^{-\alpha} + \mathcal{I}_- e^{\alpha} - \mathcal{I}_+ e^{\alpha} - \mathcal{I}_- e^{-\alpha})] \\ = \frac{e^{\alpha}+e^{-\alpha}}{2} \sum_{i=1}^{n} \sum_{l=1}^{K} \mathbf{W}_{i,j} (\mathcal{I}_+ + \mathcal{I}_-) \\ - \frac{e^{\alpha}-e^{-\alpha}}{2} \sum_{i=1}^{n} \sum_{l=1}^{K} \mathbf{W}_{i,j} (\mathcal{I}_+ - \mathcal{I}_-)\end{aligned}$$

The base learner factor α

The edge γ，描述加权重之后的正确率与错误率的差。

$$\gamma(\mathbf{W}, y, \mathbf{h}) = \Sigma_{i=1}^{m}\Sigma_{l=1}^{L}\mathbf{W}_{i,l}(\mathcal{I}_{+} - \mathcal{I}_{-})$$

同时，所有权重和为 1：$\Sigma_{i=1}^{n}\Sigma_{l=1}^{K}\mathbf{W}_{i,l}(\mathcal{I}_{+} + \mathcal{I}_{-}) = 1$

Exponential loss with γ:

$$\begin{aligned}\hat{R}_{EXP} = \tfrac{e^{\alpha}+e^{-\alpha}}{2}\sum_{i=1}^{n}\sum_{l=1}^{K}\mathbf{W}_{i,l}(\mathcal{I}_{+} + \mathcal{I}_{-})\\ -\tfrac{e^{\alpha}-e^{-\alpha}}{2}\sum_{i=1}^{n}\sum_{l=1}^{K}\mathbf{W}_{i,l}(\mathcal{I}_{+} - \mathcal{I}_{-})\\ = \tfrac{e^{\alpha}+e^{-\alpha}}{2} + \tfrac{e^{\alpha}-e^{-\alpha}}{2}\gamma.\end{aligned}$$

The base learner factor α

α minimizing the EXP loss:

$$\frac{\partial \hat{R}_{EXP}(\alpha, \mathbf{h}, \mathbf{W})}{\partial \alpha} = 0$$

$$\alpha = \frac{1}{2} \log \frac{1+\gamma}{1-\gamma}$$

The base learner factor α

α minimizing the EXP loss:

$$\frac{\partial \hat{R}_{EXP}(\alpha, \mathbf{h}, \mathbf{W})}{\partial \alpha} = 0$$

$$\alpha = \frac{1}{2} \log \frac{1+\gamma}{1-\gamma}$$

EXP loss becomes:

$$\hat{R}_{EXP}(\gamma) = \sqrt{1-\gamma^2}$$

最小化 exponential loss $\Rightarrow$ 最大化 edge。

AdaBoost.MH

input : $\mathcal{D} = (x_i, y_i) \in \{(\mathcal{R}^n, \{-1, +1\}^K\}$, T, $\mathfrak{B}$
output: $\mathbf{H} = \Sigma_{t=1}^{T} \alpha_t \mathbf{B}_t$
begin
$\mathbf{W}_1(i, l) = \frac{1}{mK}$;
for $t \leftarrow 1, \cdots, T$ **do**
Base learner and the edge: $(\mathcal{B}_t, \gamma) \leftarrow \mathfrak{B}(\mathcal{D}, \mathbf{W}_t)$;
Get hypothesis set: $h_t \leftarrow \mathbf{B}(\mathcal{D})$;
Get the base coefficient $\alpha_t \leftarrow \frac{1}{2} \log \frac{1+\gamma}{1-\gamma}$;
Update the weights:

$$\mathbf{W}_{t+1}(i, l) = \mathbf{W}_t(i, l) \exp(-\alpha_t y_{i,l} h_t(x_i)) / Z_t$$

where Z_t is a normalization factor;
end
end

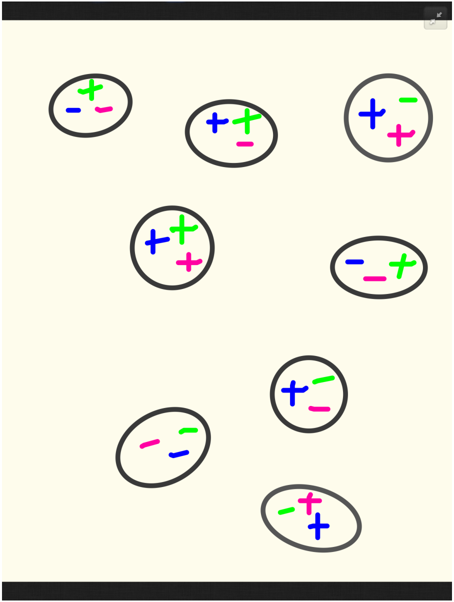
Factorization of base learners

"The return of AdaBoost.MH: multi-class Hamming trees", Kegl, 2014. 把 general vector-valued function $\mathcal{B}$ 分解

$$\mathcal{B}(\mathbf{x}) = \varphi(\mathbf{x})\mathbf{v}$$

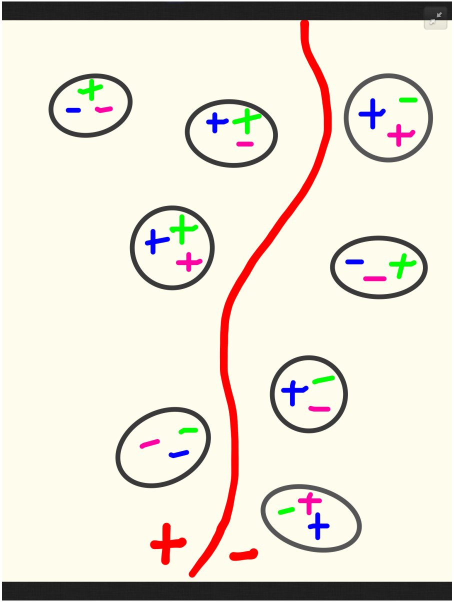
- $\varphi(x)$: a *label independent* binary classifier
  - 本质上是对 *feature space* 做划分
- $\mathbf{v}$: a *feature independent* vector-valued function
  - 将 $\varphi(x)$ 划分的 ± 1 结果 cast 到 *label space* $\{\pm 1\}^K$ 上

Factorization of base learners



有 3 个 label 的多标签分类数据集。

Factorization of base learners



$\varphi(\mathbf{x})$ 对 *feature space* 做划分，将样本集分为 $\{+1\}$ 和 $\{$-1$\}$ 两部分。

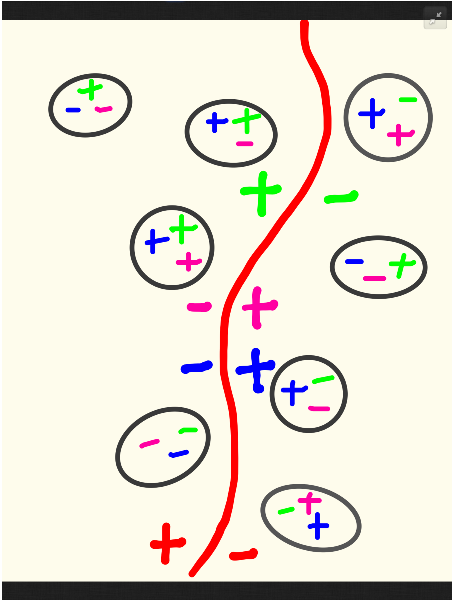
$$\gamma_{\color{green}\bullet} > 0$$

$$\gamma_{\color{blue}\bullet} < 0$$

$$\gamma_{\color{magenta}\bullet} < 0$$

$$\mathbf{v} = (v_{\color{green}\bullet}, v_{\color{blue}\bullet}, v_{\color{magenta}\bullet}) = (?, ?, ?)$$

Factorization of base learners



$\varphi(\mathbf{x})$ 正确率高的:

$v_{\color{green}\bullet} = +1,$

$\varphi(\mathbf{x})$ 错误率高的，通过 $\mathbf{v}$ 来翻转:

$v_{\color{blue}\bullet} = -1,$

$v_{\color{magenta}\bullet} = -1.$

$$\mathbf{v} = (v_{\color{green}\bullet}, v_{\color{blue}\bullet}, v_{\color{magenta}\bullet}) = (1, -1, -1)$$

最终整体的在每个 label 上，正确率都大于错误率。

AdaBoost.MH

input : $\mathcal{D} = (x_i, y_i) \in \{(\mathcal{R}^n, \{-1, +1\}^K\}$, T, $\mathfrak{B}$
output: $\mathbf{H} = \Sigma_{t=1}^{T} \alpha_t \mathbf{B}_t$
begin
$\mathbf{W}_1(i, l) = \frac{1}{mK}$;
for $t \leftarrow 1, \cdots, T$ **do**
Base learner and the edge: $((\varphi_t(\cdot), \mathbf{v}_t), \gamma) \leftarrow \mathfrak{B}(\mathcal{D}, \mathbf{W}_t)$;
Get hypothesis set: $h_t \leftarrow \mathbf{B}(\mathcal{D})$;
Get the base coefficient $\alpha_t \leftarrow \frac{1}{2} \log \frac{1+\gamma}{1-\gamma}$;
Update the weights:

$$\mathbf{W}_{t+1}(i, l) = \mathbf{W}_t(i, l) \exp(-\alpha_t y_{i,l} h_t(x_i)) / Z_t$$

where Z_t is a normalization factor;
end
end

Overview

1. 背景
   - The problem
   - User Profiling
2. 问题与求解
   - 问题抽象
   - Preliminary: Boosting
   - Multi-label boosting
3. Multiboost on Spark
   - Strong learner on Apache Spark
   - Base learner on Apache Spark
   - Base learner: generalized binary classifier with vote vector
4. 后续工作

Strong learner on Apache Spark

- Spark 的 driver program 中实现算法逻辑
- Base learner 类型作为类型参数
  - 不同 base learner 可替换，pluggable
  - Base learner 的 training 逻辑与 strong learner 解耦合

https://github.com/...../.../stronglearners/AdaBoostMH.scala

```scala
class AdaBoostMHModel[BM <: MultiLabelClassificationModel](
    numClasses: Int,
    numFeatureDimensions: Int,
    baseLearnersList: List[BM])
```

```scala
class AdaBoostMHAlgorithm[BM <: BaseLearnerModel, BA <: BaseLearnerAlgorithm[BM]](
    baseLearnerAlgo: BA,
    val numClasses: Int,
    val numFeatureDimensions: Int,
    numIterations: Int) extends StrongLearnerAlgorithm[BM, BA, AdaBoostMHModel[BM]]
```

```scala
def run(dataSet: RDD[MultiLabeledPoint]): AdaBoostMHModel[BM]
```

AdaBoost.MH on Apache Spark

迭代过程：

```
/**
 * The encapsulation of the iteration data which consists of:
 * @param model the resulted model, a strong learner, of previous iterations.
 * @param dataSet the re-weighted multilabeled data points.
 */
case class IterationData(
    model: AdaBoostMHModel[BM],
    dataSet: RDD[WeightedMultiLabeledPoint])

val finalIterationData = (1 to numIterations).foldLeft(
    IterationData(
        AdaBoostMHModel.apply[BM](numClasses, numFeatureDimensions, List()),
        weightedDataSet)) { (iterData: IterationData, iter: Int) =>
    ... 实现单次迭代的算法逻辑 ...
}
```

AdaBoost.MH on Apache Spark

```scala
// 1. train a new base learner
val baseLearner = baseLearnerAlgo.run(iterData.dataSet)
// 1.1 update strong learner
val updatedStrongLearner = AdaBoostMHModel.apply[BM](
    numClasses, numFeatureDimensions, iterData.model.models :+ baseLearner)
```

```scala
// 2. get the weak hypothesis
val predictsAndPoints = iterData.dataSet map { wmlPoint =>
  (baseLearner.predict(wmlPoint.data.features), wmlPoint)
}
```

```scala
// 3. sum up the normalize factor
val summedZ = predictsAndPoints.aggregate(0.0)({
case (sum: Double, (predict: Vector, wmlp: WeightedMultiLabeledPoint)) =>
    (predict.toArray zip wmlp.data.labels.toArray zip wmlp.weights.toArray)
        .map { case ((p, l), w) =>
          w * math.exp(-p * l)
        }.sum + sum
  }, { _ + _ })
```

```scala
// 4. re-weight the data set
val reweightedDataSet = predictsAndPoints map {
  case (predict: Vector, wmlp: WeightedMultiLabeledPoint) =>
    val updatedWeights = for (i <- 0 until numClasses)
      yield wmlp.weights(i) * math.exp(-predict(i) * wmlp.data.labels(i)) / summedZ
    WeightedMultiLabeledPoint(Vectors.dense(updatedWeights.toArray), wmlp.data)
}
```

```scala
// 5. iter data form next iteration
IterationData(updatedStrongLearner, reweightedDataSet)
```

Base learner on Apache Spark

最核心的内容是实现baseLearnerAlgo.run(iterData.dataSet)。

```scala
abstract class BaseLearnerAlgorithm[M <: BaseLearnerModel]
    extends MultiLabelClassificationAlgorithm[M] {
  def run(dataSet: RDD[WeightedMultiLabeledPoint]): M
}
```

Base learners

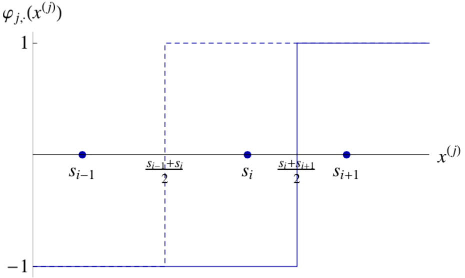
- Decision stump: 一个只有一个结点的决策树

$$\varphi_{j,b}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x}_j \geq b, \\ -1 & \text{otherwise.} \end{cases}$$

- Hamming tree: Decision stump 作为结点的决策树
- Generalized bin-classifier 方案
  - $\varphi(x)$ 使用任意二分类模型，与 $\mathbf{v}$ 一起来最大化 class-wise edge/最小化 exp loss。

Decision stump model

- 寻找最优划分来 maximize "discriminability": $(j*, b*) = \arg\max_{j,b} \sum_l \|\gamma_l\|$



  - If $y_{i,l} = 1, \gamma_l - = 2w_{il}$; Or $y_{i,l} = -1, \gamma_l + = 2w_{i,l}$.
- 单机版： $\mathbf{x}_j$ 排序复杂度 $\mathcal{O}(m\log(m))$，搜索的过程 $\mathcal{O}(m)$，总的 $\mathcal{O}(n(m\log(m) + m))$
- Spark: flatMap $\rightarrow$ reduceByKey

Decision stump on Spark: Implementation

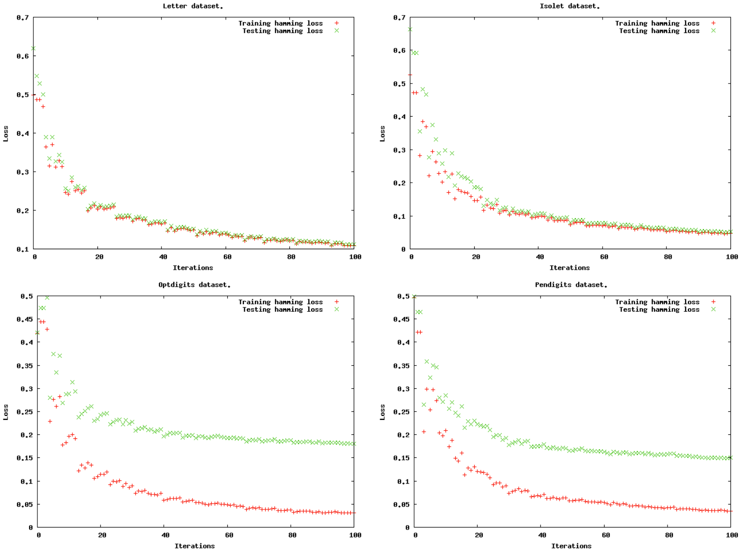
```scala
/**
 * The data abstraction of feature split metrics.
 * @param featureCut the feature and the split value of the stump
 * @param edges the vector of class-wise edges of the stump
 */
case class SplitMetric(featureCut: Option[FeatureCut], edges: Vector) extends Serializable

/**
 * Inside each partition, evaluate the local split metrics.
 * @param featureSet a set of the selected feature index
 * @return local split metrics
 */
def getLocalSplitMetrics(featureSet: Iterator[Int])(
  dataSet: Array[WeightedMultiLabeledPoint]): Iterator[SplitMetric]

/**
 * For each split, aggregate the edges from different data partitions.
 * @param allSplitMetrics RDD of all split candidates and the corresponding metrics
 * @return the summed metrics for each split candidate
 */
def aggregateSplitMetrics(allSplitMetrics: RDD[SplitMetric]): RDD[SplitMetric]

/**
 * Find the best stump split to minimize the loss given all split candidates.
 * @param splitMetrics the collection of split candidates and corresponding edges
 * @return the best feature cut
 */
def findBestSplitMetrics(splitMetrics: RDD[SplitMetric]): DecisionStumpModel
```

Decision stump on Spark: Results



"letter", "isolet", "optdigits" and "pendigits" from UCI dataset

http://archive.ics.uci.edu/ml/datasets.html

Decision stump on Spark: Performance

n: 150k+, $\|\mathbf{D}\| = 342926$; run on 500 RDD partitions, 150 executors.

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Shuffle Read	Shuffle Write
27	aggregate at AdaBoostMH.scala:139	2014/09/01 18:16:20	2 s	518/518	270.6 MB	
24	aggregate at DecisionStump.scala:161	2014/09/01 18:16:11	10 s	518/518	11.6 GB	
25	reduceByKey at DecisionStump.scala:145	2014/09/01 18:10:27	5.7 min	518/518	270.7 MB	11.6 GB
22	aggregate at AdaBoostMH.scala:139	2014/09/01 18:10:21	6 s	518/518	270.7 MB	
19	aggregate at DecisionStump.scala:161	2014/09/01 18:10:13	8 s	518/518	11.5 GB	
20	reduceByKey at DecisionStump.scala:145	2014/09/01 18:04:24	5.8 min	518/518	270.6 MB	11.6 GB
17	aggregate at AdaBoostMH.scala:139	2014/09/01 18:04:18	6 s	518/518	270.6 MB	
14	aggregate at DecisionStump.scala:161	2014/09/01 18:04:09	9 s	518/518	11.5 GB	
15	reduceByKey at DecisionStump.scala:145	2014/09/01 17:58:34	5.6 min	518/518	270.7 MB	11.5 GB

Base learner: generalized binary classifier with vote vector

- Decision stump 的问题
  - 是非常弱的二分类模型
  - Decision stump 模型训练的数据传输量很大
  - Tree-based 模型，并不适合高维稀疏数据

Base learner: generalized binary classifier with vote vector

- Decision stump 的问题
  - 是非常弱的二分类模型
  - Decision stump 模型训练的数据传输量很大
  - Tree-based 模型，并不适合高维稀疏数据

我们需要一个更强的、更易于训练并且适应高维稀疏数据的 $\varphi(\cdot)$，来对 *feature space* 做二元划分。

Generalized binary φ on Apache Spark

- 使用 spark.mllib.classification 中已有的 model/algorithm
  - SVM, LR with GradientDescent, LBFGS.

```
class SVMClassificationModel(svmModel: SVMModel)
    extends BinaryClassificationModel with Serializable

class SVMClassificationAlgorithm
    extends BinaryClassificationAlgorithm[SVMClassificationModel]
    with Serializable
```

- Boosting 机制，只要 base learner 比 random guess 好，整体就是收敛的
  - 通过 vote vector $\mathbf{v}$ 来保证每个 label 上的错误率小于 0.5。
- 将 $\mathbf{D} \in \{(\mathcal{R}^n, \{\pm 1\}^K)\}$ 转化成 $\mathbf{D}' \in \{(\mathcal{R}^n, \{\pm 1\})\}$，作为 binary classification 的数据集
  - 使用 $\{\pm 1\}^K$ 中的一个维度作为 1 维 label 的取值
  - $\mathbf{D} \Rightarrow \mathbf{D}'$ 是需要进一步考虑的部分 *TODO

后续工作

- 训练 $\varphi(\cdot)$ 时，如何转换数据集 $\mathbf{D} \Rightarrow \mathbf{D}'$，使得 base learner 的效果最好
- 使用 LR、SVM 作为 $\varphi(\cdot)$，其 loss 与 edge-maximizing 目标一致性需要理论证明
- $\star$ 现有实现基于 Spark 1.1.x, 应用新的 spark ml interface
- $\star\star\star$ 定义新的判别模型 $\varphi(\cdot)$，实现 exp loss 对应的梯度计算 $\frac{\partial \hat{R}_{EXP}}{\partial \varphi(\cdot)}$，优化求解模型参数

欢迎 fork & pull request!

https://github.com/baigang/spark_multiboost