Pattern Recognition

Lecture 11

Ensemble Learning

主要内容

- 什么是分类器集成
- 为什么要分类器集成
- 集成学习成功的要素
- 典型的集成学习算法介绍
- 选择性集成

什么是分类器集成

- 狭义：利用多个同质的分类器共同解决一个问题，“同质”指的是分类器所属同一种类型，例如都是决策树，神经网络等
- 广义：只要使用多个分类器即可，可以不同质，也称作多分类器系统。
- 关键问题：获得一组好的分类器+将他们以一种合适的方式“集成”起来

如何集成分类器

- 训练好组件分类器后，**最常用**的集成方法是majority voting
- Weighted majority voting
- 其他各种集成方法

为什么要分类器集成

- 降低只选择单个分类器的风险
- 集成的分类器可以获得更好的性能
- 以神经网络为例
  - 训练分类器就是是寻找最优权值的过程，这是一个优化问题，有很多局部最优
  - 不同的训练过程（初始点，输入数据的顺序等不同），得到的神经网络，在输入空间的不同子集上的表现不同
  - 相比于由单个神经网络做出的决策，由多个神经网络共同做出的决策错误的可能性低

集成学习成功的要素

- ensemble 中的每个分类器都具有足够好的性能
- ensemble 中的各个分类器之间是有差异的

- 单个分类器的性能可以通过对个体分类器进行充分训练获得，那么如何获得差异性？
  - 关键在于：如何使得训练的过程有所不同

集成学习成功的要素

- 单个分类器的训练总的来说由两大基本要素确定：训练样本和训练算法
  - 使用不同的训练样本
  - 使用不同的训练算法
    - 分类器采用不同的模型（如神经网络、决策树、SVM）
    - 采用相同的模型，但是使用不同的初始参数进行训练（多用于随机性比较强的训练算法）
    - 在训练过程中使用不同的参数，如学习速率

典型的集成学习算法-Bagging

- Bagging （ bootstrap aggregating ）
  - 基本思路：利用不同的样本集合训练单个分类器
  - 令原始数据集为D (n个样本)，步骤如下：

算法1（Bagging）

1.for i=1 to T

2. 从D中独立随机抽取m个数据(m<n)，形成数据集S_i

3. 用S_i训练得到一个分类器

4.end

5.所有分类器的判决结果投票决定最终的分类判决

典型的集成学习算法-Bagging

- Bagging 主要是减少分类模型的方差(variance)
- 因此，Bagging 一般用于训练算法对于训练数据较为敏感（即unstable）的场合（例如：判别树）
- Bagging 在分类问题中，也会得到较差的结果
  - 例：
    - 假设对所有输入样本，其真实的label都是A
    - Bagging中的单个分类器对所有样本，全部以0.4的概率分成A，0.6的概率分成B，则单个分类器的误差为0.6 (unstable)
    - 最终Bagging的误差为1

典型的集成学习算法-Boosting

- Boosting
  - 按照一定的方法从原始的训练样本集D中选取一个子集S
  - 利用S训练一个弱学习器，该学习器对S上的数据的预测只要比随机猜测的准确率高一点即可
  - 更新选取样本的规则
  - 重复以上过程得到若干分类器
  - 采用投票方法决定最后的分类预测

典型的集成学习算法-Boosting

- AdaBoost
  - **基本思路：**

串行地训练一组分类器，使它们逐渐“聚焦于”比较困难的样本上。

  - **具体地说：**

每一个样本都被赋予一个权重，表明其被选为训练集的概率

修改权重的规则：如果某样本已被正确分类，则降低权重，否则提高权重。

最后根据每个分量分类器的性能，以加权投票的方式进行决策

典型的集成学习算法-Boosting

算法 3（AdaBoost）

1. **Begin initialize** $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$, k_{max}, $W_1(i) = \frac{1}{n}, i = 1, \dots, n$
2. $k = 0$
3. **do** $k \leftarrow k + 1$
4. 训练使用$W_k(i)$采样的D的弱分类器C_k
5. $E_k \leftarrow$对使用$W_k(i)$的D测量的C_k的训练误差
6. $\alpha_k \leftarrow \frac{1}{2}\ln[(1 - E_k)/E_k]$
7. $W_{k+1}(i) = \frac{W_k(i)}{Z_k} \times \begin{cases} e^{-\alpha_k} \text{如果} h_k(x_i) = y_i \\ e^{\alpha_k} \text{ 如果} h_k(x_i) \neq y_i \end{cases}$
8. **until** $k = k_{max}$
9. **return** C_k 和 $\alpha_k, k = 1, \dots, k_{max}$
10. **end**

典型的集成学习算法-Boosting

- $h_k(x)$是分类器C_k在x上给出的类别号
- 最后的判别函数为：

$$\text{sign}[g(x)] = sign\left[\sum_{k=1}^{k_{max}} \alpha_k h_k(x)\right]$$

- 一个关于boosting的网站：www.boosting.org
- 此外，AdaBoost还有很多变体。具体实现见WEKA等工具

典型的集成学习算法-NCL

- NCL
  - 基本思想：
    - ensemble成功的要素：Accuracy和Diversity
    - NCL同时训练*M*个BP神经网络
    - 训练过程中使各个神经网络之间具有负相关性(Diversity)，并且保证每个神经网络较高的分类精度(Accuracy)
    - NCL的输出取所有网络的平均，即

$$f = \frac{1}{M}\sum_{i=1}^{M} f_i$$

典型的集成学习算法-NCL

- BP算法中，第n个样本的误差函数定义为

$$E(n) = \frac{1}{2}\left(f(n) - t(n)\right)^T\left(f(n) - t(n)\right)$$

其中，$f(n)$为网络的输出，$t(n)$为真实的类别

- NCL中，误差函数加入相关惩罚项，定义第i个网络在第n个样本上的误差函数为

$$E_i(n) = \frac{1}{2}\left(f_i(n) - t(n)\right)^T\left(f_i(n) - t(n)\right) + \lambda p_i(n)$$

典型的集成学习算法-NCL

- $p_i(n)$为一个惩罚项，要求各个网络的输出各不相同，具体定义为

$$p_i(n) = (f_i(n) - f(n))^T \sum_{j \neq i} (f_j(n) - f(n))$$

- 误差函数对权值求偏导

$$\frac{\partial E_i(n)}{\partial w} = \frac{\partial E_i(n)}{\partial f_i(n)} \cdot \frac{\partial f_i(n)}{\partial w}$$

$$= \left[\frac{1}{2} \frac{\partial ((f_i(n) - t(n))^T (f_i(n) - t(n)))}{\partial f_i(n)} + \lambda \frac{\partial p_i(n)}{\partial f_i(n)} \right] \cdot \frac{\partial f_i(n)}{\partial w}$$

$$= \left[(f_i(n) - t(n)) + \lambda \sum_{j \neq i} ((f_j(n) - f(n))) \right] \cdot \frac{\partial f_i(n)}{\partial w}$$

$$= \left[(f_i(n) - t(n)) - \lambda (f_i(n) - f(n)) \right] \cdot \frac{\partial f_i(n)}{\partial w}$$

典型的集成学习算法-NCL

- 负相关学习是对标准BP算法的扩展，实际上就是在误差反向传播时额外计算一项$\lambda(f_i(n)-f(n))$

算法 4（NCL）

1. **Begin initialize** 训练数据 $D=\{(x_1,d_1),\ldots,(x_n,d_n)\}$，分类器数$M$
2. for $t=1$ to T
3. for $j=1$ to n
4. 计算$\bar{f}=\frac{1}{M}\sum_i f_i(x_j)$
5. **for** i=1 to M
6. 对第i个神经网络的权值进行更新
7. $\Delta w=-\alpha[(f_i(x_n)-d_n)-\lambda(f_i(x_n)-\bar{f})]\cdot\frac{\partial f_i}{\partial w}$
8. **end**
9. **end**
10. **end**

典型的集成学习算法-Stacking

- Stacking （也称stacked generalization）
  - 主要思想：两层的集成学习
    - 第一层：用多个异构的分类算法在原始的数据集上训练，得到多个分类器
    - 用第一层分类器的输出作为输入，训练第二层的分类器

典型的集成学习算法-Stacking

算法 2（Stacking）

数据D，第一层训练算法$L_1, L_2, \ldots, L_T$，第二层训练算法 L'

1. for i=1 to T:

2. $h_t = L_i(D)$

3. end

4. $D' = \emptyset$

5. for j=1 to m

6. **for** t=1 to T

7. $z_{it} = h_t(x_i)$

8. **end**

9. $D' = D' \cup \{((z_{i1}, z_{i2}, \ldots, z_{iT}), y_i)\}$

10.end

11.$h' = L'(D')$

12.输出：$H(x) = h'(h_1(x), \ldots, h_T(x))$

典型的集成学习算法

- Ensemble 算法比较
  - Bagging：分量分类器的训练过程互相独立，可同时进行，适用于unstable的分类器，一般通过投票法进行决策
  - Stacking：第一层分量分类器相互独立，第二层和第一层之间串行方式进行
  - AdaBoosting：分量分类器以串行的方式训练，后训练的分类器依赖于先训练出的分类器的性能，适用于几乎所有训练算法，一般通过投票法进行决策
  - NCL：分量分类器的训练过程相关，可同时进行，但暂时仅适用于训练神经网络

选择性集成

- 分类器集成通过利用多个分类器来获得比及使用单个分类器更强的泛化能力。
- 尽管如此，分类器也并非越多越好，因为
  1. 更多的分类器导致更大的计算和存储开销
  2. 越多的分类器会使个体之间的差异也越来越难获得
  3. 实验证明，过多的分类器有可能降低集成的泛化能力
- 选择性集成(Selective Ensemble)
  并不使用所有训练好的分类器，而是从已有的个体分类器中进行选择之后再集成。
- 目标：获得更好的泛化能力，同时降低存储和计算开销

选择性集成

- 考虑两类分类问题，设目标输出为

$$\boldsymbol{t} = (t_1, t_2, \ldots, t_N)$$

第i个分类器的实际输出为

$$\boldsymbol{F}_i = (f_{i1}, f_{i2}, \ldots, f_{iN})$$

其中$t_j, f_{ij} \in \{-1, +1\}$

- 采用投票的方法进行集成，可定义集成在第j个样本上的输出

$$O_j = sign(sum_j)$$

其中，$sum_j = \sum_{i--=1}^{M} f_{ij}$，$sign(x) = \begin{cases} 1\ if\ x > 0 \\ 0\ if\ x = 0 \\ -1\ if\ x < 0 \end{cases}$

选择性集成

- 定义函数

$$Error(x) = \begin{cases} 1, & if\ x = -1 \\ 0.5, & if\ x = 0 \\ 0, & if\ x = 1 \end{cases}$$

- 则集成的分类错误率为

$$E = \frac{1}{N}\sum_{j=1}^{N} Error(O_j t_j)$$

选择性集成

- 现对集成进行筛选，假设将第k个分类器从集成中删除，则新的集成的错误率为

$$E' = \frac{1}{N}\sum_{j=1}^{N} Error(Sgn(sum_j - f_{kj})t_j)$$

- 只要满足$E' \le E$就说明删除第k个分类器后的集成比原来的集成好，即

$$\sum_{j=1}^{N}\{Error(Sgn(sum_j)t_j) - Error(Sgn(sum_j - f_{kj})t_j)\} \ge 0$$

选择性集成

- 当$|sum_j| > 1$时，删除第k个分类器对集成在第j个样本上的的输出没有影响

- $|sum_j| \le 1$ 时，$sum_j \in \{-1, 0, +1\}$，有

$$Error(Sgn(sum_j)t_j) - Error(Sgn(sum_j - f_{kj})t_j) = -\frac{1}{2}Sgn(sum_j + f_{kj})t_j$$

- 若删除后的集成更好，则有

$$\sum_{\substack{j=1 \\ j \in \{j \,||sum_j| \le 1\}}}^{N} Sgn((sum_j + f_{kj})t_j) \le 0$$

- 现可以找到一种情况，满足以上条件，所以选择行集成是可行的