Machine Learning-I (CS/DS 706)

End Semester Exam

Prof.G.Srinivasaraghavan

Date Posted: Nov 27, 2017	**Submit By**: Dec 10, 2017, Midnight	**Max. Marks**: 100

1. **Implementation (R Or Python)**: Implement a variant of the Feature Weighted Linear Stacking (FWLS) scheme for producing ensembles from a collection of heterogeneous models for multiclass classification. This was very closely related to the scheme that was used in the winning solution for the famous \$1 Million Netflix Challenge few years ago. You may want to refer to `https://arxiv.org/pdf/0911.0460.pdf`. The ensembling scheme is described below – in fact we will sample several from a family of ensembles and choose one using cross-validation. You are given a collection of learning algorithms $\mathcal{A}_1, \ldots \mathcal{A}_m$ for classification (think Logistic regression, SVM, random forest, ...) and a *blender* algorithm $\mathcal{B}$ that blends these base algorithms into a higher level ensemble. $\mathcal{B}$ is another familiar classification algorithm, including one of the algorithms already in the 'base' list. You are also given an array of 'preferences' $(p_1, \ldots, p_m)$ among the base algorithms — higher the p_i more $\mathcal{B}$ will be biased towards $\mathcal{A}_i$. Finally you are also given the total number of base models N you want to use (you can generate multiple models using the same algorithm — we do this by changing the hyperparameters every time).

   Let $\mathcal{D} = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$ where $y_i \in \{1, \ldots, C\}$ are the class labels and $x_{i1}, \ldots, x_{id}$ are the d features associated with datapoint $(\boldsymbol{x}_i, y_i)$, be the dataset. We do a k-fold cross validation on the training data. Let the 'folds' in the training data for cross validation be denoted $F_1, \ldots, F_k$ — note that $|F_i| = n/k$ (we assume n is divisible by k) and let F_{-i} denote all the datapoints other than those in F_i. We assume that each of the base algorithms result in models that will produce a distribution over the C classes when tested on any data point $\boldsymbol{x}$ (you may have a do a little bit of tweaking and/or make appropriate use of some of less used parameters in the functions that implement the base algorithms in R/Python).

   Since the blending algorithm $\mathcal{B}$ will be working with a large number of dimensions. it is preferably something that will scale well with dimensions. Preferred choice for this will be random forest or some variant like boosted trees (*xgboost*).

   The meta-algorithm to create the ensemble is shown below (1) — the top-level function is the *BlendingEnsemble* function. Note that the pseudo-code shown does not necessarily reflect the code structure — the way it is shown here is more for explanation of what the code is supposed to do. To develop your code in Python the *estimator* interface and the *GridSearchCV/RandomizedSearchCV* functions in SciKit-Learn might be worth looking at. Similarly in R you could take a look at the *caret, EnsembleBase* packages.
   Max Marks: 50

2. Attributes for a regression problem can also be noisy. To bring in some robustness to potential attribute noise suppose we assume that the training data points $(\boldsymbol{x}_i, y_i), 1 \leq i \leq n$ are such that each $\boldsymbol{x}_i$ is of the form $\hat{\boldsymbol{x}}_i + \boldsymbol{\epsilon}_i$ where $\hat{\boldsymbol{x}}_i$ is the 'true' measurement and $\boldsymbol{\epsilon}_i$ is a zero mean noise vector with covariance matrix $\sigma^2 I$. Assume that the noise vectors $\boldsymbol{\epsilon}_i$ are all i.i.d. We now formulate the standard linear regression problem slightly differently — instead of minimizing the sum of the square errors we minimize the expectation (w.r.t the attribute noise) of the cumulative square error. Derive the solution to this modified form of linear regression and show that it leads to a simple generalized form of ridge regression. **Max Marks: 15**

Algorithm 1 Blending Ensemble

```
procedure GenParams($\mathcal{A}_1, \ldots \mathcal{A}_m, \mathcal{B}, P, N$)    ▷ $P$-Preference Array; $N$-Number of models
    $(\mu_1, \ldots, \mu_m)$ = Sample a distribution from a Dirichlet with parameters $P$
    $N_i = \mu_i * N, \quad 1 \le i \le m$    ▷ We Will generate $N_i$ models from $\mathcal{A}_i$
    $\boldsymbol{\Psi}_{ij}$ = Sample hyperparameters for $\mathcal{A}_i$ from an appropriate distribution, $1 \le i \le m, 1 \le j \le N_i$
    $\boldsymbol{\Phi}$ = Sample hyperparameters for $\mathcal{B}$
    $\boldsymbol{N} = (N_i \mid 1 \le i \le m)$; $\boldsymbol{\Psi} = (\boldsymbol{\Psi}_{ij} \mid 1 \le i \le m, 1 \le j \le N_i)$
    return $\boldsymbol{\Phi}, \boldsymbol{N}, \boldsymbol{\Psi}$
end procedure

procedure Blend($\mathcal{A}_1, \ldots \mathcal{A}_m, \mathcal{B}, D, L, \boldsymbol{\Phi}, \boldsymbol{N}, \boldsymbol{\Psi}$)
    Set $\rho = 0.7$    ▷ This can be set to an appropriate value in the range $[0, 1]$
    $D_{fw} = \phi$    ▷ A new feature weighted dataset
    for $l = 1, \ldots, L$ do
        $D'$ = Sample from $D$ of size $\rho * |D|$; $\overline{D'} = D - D'$
        for $i = 1, \ldots, m; j = 1, \ldots, N_i$ do
            $M_{ij} = \mathcal{A}_i(D' \mid \boldsymbol{\Psi}_{ij})$    ▷ Run $\mathcal{A}_i$ on $D'$ with hyperparameters $\boldsymbol{\Psi}_{ij}$ to get model $M_{ij}$
        end for
        for $(\boldsymbol{x}, y) \in \overline{D'}$ do    ▷ For every 'test' point $\boldsymbol{x}$
                                              ▷ Note that $M_{bc}(\boldsymbol{x})$ generates a probability vector of length $C$
            Construct $\boldsymbol{F} = \odot(\boldsymbol{F}_{abc})$ where $\boldsymbol{F}_{abc} = x_a * (M_{bc}(\boldsymbol{x}))$
            Construct $\boldsymbol{G} = \odot(\boldsymbol{G}_{ij})$ where $\boldsymbol{G}_{ij} = (M_{ij}(\boldsymbol{x}))$    ▷ $\odot$ is the concatenation operator
            $D_{fw} = D_{fw} \cup \{(\boldsymbol{x}, \boldsymbol{G}, \boldsymbol{F}, y)\}$
                  ▷ This is the new feature vector consisting of the original features, the base model
                  ▷ predictions and pairwise products of the base model predictions and the features
        end for
    end for
    $M = \mathcal{B}(D_{fw} \mid \boldsymbol{\Phi})$ ▷ Run $\mathcal{B}$ on $D_{fw}$ with hyperparameters $\boldsymbol{\Phi}$ to get the final blended model $M$
    return $M$
end procedure

procedure BlendingEnsemble($\mathcal{A}_1, \ldots \mathcal{A}_m, \mathcal{B}, \mathcal{D}, P, N, k$)    ▷ $k$-# folds for cross validation
    $L = 2$
    for $r = 1, \ldots, R$ do
                                              ▷ Searching over hyperparameters
        $\boldsymbol{\Phi}_r, \boldsymbol{N}_r, \boldsymbol{\Psi}_r = GenParams(\mathcal{A}_1, \ldots \mathcal{A}_m, \mathcal{B}, P, N)$
                                              ▷ Standard Cross Validation for each choice of hyperparameters
        for $t = 1, \ldots, k$ do
            $M_t = Blend(\mathcal{A}_1, \ldots \mathcal{A}_m, \mathcal{B}, F_{-t}, L, \boldsymbol{\Phi}_r, \boldsymbol{N}_r, \boldsymbol{\Psi}_r)$
        end for
        $\epsilon_r = Mean\left(\{Error(M_t(F_t)) \mid 1 \le t \le k\}\right)$
    end for
    $r^* = \arg\min\left(\{\epsilon_r \mid 1 \le r \le R\}\right)$
                  ▷ Output the model with the best performance — Build it on the entire dataset
    $M = Blend(\mathcal{A}_1, \ldots \mathcal{A}_m, \mathcal{B}, \mathcal{D}, L, \boldsymbol{\Phi}_{r^*}, \boldsymbol{N}_{r^*}, \boldsymbol{\Psi}_{r^*})$
    return $M$
end procedure
```

3. Show that the VC dimension $d_{\mathcal{H}}$ of any finite hypothesis space $\mathcal{H}$ is at most $\log_2 \mathcal{H}$. Show that this bound is tight — for any $d > 1$, there exists a hypothesis class $\mathcal{H}$ such that $d = d_{\mathcal{H}}$. **Max Marks: 5**

4. Assume you are given class conditional probabilities for each of the attribute values — $\theta_{ijk} = p(x_i = v_{ij} \mid y = k)$ is the probability that i^{th} attribute of the data point $(\boldsymbol{x}, y)$ takes the j^{th} value (among the distinct values $v_{i1}, \ldots, v_{i,m_i}$ that the i^{th} attribute can take) given that the class label is k. Write the pseudocode for a decision tree building scheme that (i) uses Information Gain, (ii) the χ^2 stopping criterion, (iii) can handle missing values, and (iv) gives out class membership probabilities instead of class 'decisions'. Also write the pseudocode for how such a tree will give out a distribution over class membership for any given test point. **Max Marks: 15**

5. Show how you can determine (using a kernel) the center of mass $\boldsymbol{\mu}$ in the feature space and the average of the square Euclidean distances from $\boldsymbol{\mu}$ to each $\boldsymbol{\phi}(\boldsymbol{x})$ where $\boldsymbol{x}$ is the original feature vector and $\boldsymbol{\phi}$ is the feature transformation. **Max Marks: 15**