Ensemble Learning

Intro

- We introduce **ensemble classifiers**, that is, machine learning based classifiers that utilize a combination of scoring functions

Intro

- In **ensemble learning**, multiple learning algorithms are combined, with the goal of improved accuracy as compared to the individual algorithms

  Ensemble techniques are widely used, and as a testament to their strength, ensembles have won numerous machine learning contests in recent years, including the KDD Cup, the Kaggle competition, and the Netflix prize

  - Unfortunately, they aim to accuracy more than efficiency, and in real-world systems, practicality and efficiency are necessarily crucial factors

Ensemble Classifiers

- There are various ways to combine classifiers, for example straightforward combinations, such as a **maximum**, **sum**, **product**, **majority vote**, and so on

- These approaches are classic, we will see also something “different”

A Framework for Ensemble Classifiers

We consider ensemble learners that are based on combinations of scoring functions

- Note that we place no additional restrictions on the scoring functions and, in particular, they do not necessarily represent "learning" algorithms, per se.

- Hence, we are dealing with **ensemble methods** broadly speaking, rather than ensemble learners in a strict sense

  We assume that the ensemble method itself—as opposed to the scoring functions that comprise the ensemble—is for classification, and hence ensemble functions are 0-1 loss functions

In a 0-1 loss function you simply count the number of misclassified items

Notation

- The score generated by a **scoring function** $\mathcal{S}$ when applied to **sample** x is given by:

$$\mathcal{S}(x;\ V,\ \Lambda)$$

where we have explicitly included the dependence on the **training data** V and the **function parameters Λ**

Notation

- In its most general form, an ensemble method for a binary classification problem can be viewed as a function $F : R^{\ell} \rightarrow \{0, 1\}$ of the form:

$$F\left(\mathcal{S}_1(x; V_1, \Lambda_1), \mathcal{S}_2(x; V_2, \Lambda_2), \ldots, \mathcal{S}_{\ell}(x; V_{\ell}, \Lambda_{\ell})\right)$$

- That is, the ensemble method defined by the function F produces a classification based on the scores $\mathcal{S}_1,\ \mathcal{S}_2,\ \ldots,\ \mathcal{S}_{\ell}$

➢Where scoring function $\mathcal{S}_i$ is trained using the **data** $\boldsymbol{V_i}$ and **parameters** $\boldsymbol{\Lambda_i}$

Classifying Ensemble Classifiers

- From a high-level perspective, ensemble classifiers can be categorized as **bagging**, **boosting**, **stacking**, or some combination thereof

Bagging

- In **bootstrap aggregation** (i.e., **bagging**), different subsets of the data or features (or both) are used to generate different scores

  The results are then combined in some way, such as a sum of the scores, or a majority vote of the corresponding classifications

- For bagging we assume that the same scoring method is used for all scores in the ensemble.

- One benefit of bagging is that it reduces overfitting

Bagging

For bagging, the general equation is restricted to:

$$F\big(\mathcal{S}(x; V_1, \Lambda), \mathcal{S}(x; V_2, \Lambda), \ldots, \mathcal{S}(x; V_\ell, \Lambda)\big)$$

- That is, in bagging, each scoring function is essentially the same, but each is trained on a different feature set

Sample 1 Sample 2 Sample 3

Matrix V

Feature 1	Feature 1	Feature 1
Feature 2	Feature 2	Feature 2
Feature 3	Feature 3	Feature 3
Feature 4	Feature 4	Feature 4
Feature 5	Feature 5	Feature 5
Feature 6	Feature 6	Feature 6
Feature 7	Feature 7	Feature 7

- For example, suppose that we collect all available feature vectors into a matrix V
- Then bagging based on subsets of samples would correspond to generating V_i by deleting a subset of the columns of V
- On the other hand, bagging based on features would correspond to generating V_i by deleting a subset of the rows of V

Boosting

- **Boosting** is a process whereby **distinct classifiers are combined** to produce a stronger classifier

Generally, boosting deals with weak classifiers that are combined in an adaptive or iterative manner to improve the overall classifier

- We restrict our definition of boosting to cases where the classifiers are closely related, in the sense that they differ only in terms of parameters

Boosting

- From this perspective, boosting can be viewed as “bagging” based on classifiers, rather than data or features
  - That is, all of the scoring functions are reparametrized versions of the same scoring technique

Under this definition of boosting, the general equation becomes:

$$F\big(\mathcal{S}(x;V,\Lambda_1),\mathcal{S}(x;V,\Lambda_2),\ldots,\mathcal{S}(x;V,\Lambda_\ell)\big)$$

  - That is, the scoring functions differ only by re-parameterization, while the **scoring data and features do not change**
- We will see a specific examples of boosting, that is, the most popular method known as **AdaBoost**

Stacking

- **Stacking** is an ensemble method that combines disparate scores using a **metaclassifier**

  In this generic form, stacking is defined by the general case in equation, where the scoring functions can be (and typically are) significantly different

$$F\left(\mathcal{S}_1(x; V_1, \Lambda_1), \mathcal{S}_2(x; V_2, \Lambda_2), \ldots, \mathcal{S}_\ell(x; V_\ell, \Lambda_\ell)\right)$$

- Note that from this perspective, stacking is easily seen to be a generalization of both bagging and boosting

  Because stacking generalizes both bagging and boosting, it is not surprising that stacking based ensemble methods can outperform bagging and boosting methods

Stacking

- However, this is not the end of the story, because stacking introduces a large overhead

- Of course, the appropriate tradeoffs will depend on the specifics of the problem at hand

Ensemble Methods

- Bagging: $F\big(\mathcal{S}(x; V_1, \Lambda), \mathcal{S}(x; V_2, \Lambda), \ldots, \mathcal{S}(x; V_\ell, \Lambda)\big)$

  Same Scoring function with the same parameters Λ, but training data/features wary

- Boosting: $F\big(\mathcal{S}(x; V, \Lambda_1), \mathcal{S}(x; V, \Lambda_2), \ldots, \mathcal{S}(x; V, \Lambda_\ell)\big)$

  Same Scoring function with the same training data/features, but different parameters Λ

- Stacking $F\big(\mathcal{S}_1(x; V_1, \Lambda_1), \mathcal{S}_2(x; V_2, \Lambda_2), \ldots, \mathcal{S}_\ell(x; V_\ell, \Lambda_\ell)\big)$

  Totally different Scoring functions combined via a *meta-classifier*

Ensemble Classifier Examples

- Here, we consider a variety of ensemble methods

- We begin with a few generic examples, and then discuss several more specific examples.

Maximum

In this case, we have:

$$F\big(\mathcal{S}_1(x;V_1,\Lambda_1),\mathcal{S}_2(x;V_2,\Lambda_2),\ldots,\mathcal{S}_\ell(x;V_\ell,\Lambda_\ell)\big)=\max\{\mathcal{S}_i(x;V_i,\Lambda_i)\}$$

Averaging

In this case, we have:

$$F\left(\mathcal{S}_1(x;V_1,\Lambda_1),\mathcal{S}_2(x;V_2,\Lambda_2),\ldots,\mathcal{S}_\ell(x;V_\ell,\Lambda_\ell)\right) = \frac{1}{\ell}\sum_{i=0}^{\ell}\mathcal{S}_i(x;V_i,\Lambda_i)$$

Voting

- Voting could be used as a form of **boosting**, provided that no bagging is involved (i.e., the same data and features are used in each case)
- Voting is also applicable to **stacking**, and in the case of stacking, a simple majority vote is of the form:

$$F\big(\widehat{\mathcal{S}}_1(x;V_1,\Lambda_1),\widehat{\mathcal{S}}_2(x;V_2,\Lambda_2),\ldots,\widehat{\mathcal{S}}_\ell(x;V_\ell,\Lambda_\ell)\big)$$
$$= \mathrm{maj}\big(\widehat{\mathcal{S}}_1(x;V_1,\Lambda_1),\widehat{\mathcal{S}}_2(x;V_2,\Lambda_2),\ldots,\widehat{\mathcal{S}}_\ell(x;V_\ell,\Lambda_\ell)\big)$$

where “maj” is the majority vote function

- Note that the majority vote is well defined in this case, provided that ℓ is odd—if ℓ is even, we can simply flip a coin in case of a tie

Voting

- As an aside, we note that it is easy to see why we want to avoid correlation when voting is used as a combining function

  Suppose that we have the three highly correlated scores:

$$\begin{pmatrix} \widehat{\mathcal{S}}_1 \\ \widehat{\mathcal{S}}_2 \\ \widehat{\mathcal{S}}_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

  where each 1 indicates correct classification, and each 0 is an incorrect classification

  - Then, bot $\mathcal{S}_1$ anŇ $\mathcal{S}_2$ are 80% accurate, anŇ $\mathcal{S}_3$ is 70% accurate. If we use a simple majority vote, then we obtain the classifier

$$C = (\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ )$$

    which is 80% accurate

Voting

- On the other hand, the <u>less correlated classifiers</u>:

$$\begin{pmatrix} \widehat{\mathcal{S}}_1' \\ \widehat{\mathcal{S}}_2' \\ \widehat{\mathcal{S}}_3' \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

are only 80%, 70% and 60% accurate, respectively, but the majority vote in this case gives us

$$C' = (\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ )$$

which is 90% accurate

ML-Based Combination

- Recall that the most general formulation of an ensemble classifier is given by:

$$F\left(\mathcal{S}_1(x; V_1, \Lambda_1), \mathcal{S}_2(x; V_2, \Lambda_2), \ldots, \mathcal{S}_\ell(x; V_\ell, \Lambda_\ell)\right)$$

- In this formulation, we can select the function F based on a machine learning technique

SVM as Meta-Classifier

- It is natural to use an SVM as a meta-classifier to combine scores

- The use of SVM in this meta-classifier mode can be viewed as a general **stacking** method

  Thus, this SVM technique is equivalent to the equation:

$$F\left(\mathcal{S}_1(x;V_1,\Lambda_1),\mathcal{S}_2(x;V_2,\Lambda_2),\ldots,\mathcal{S}_\ell(x;V_\ell,\Lambda_\ell)\right)$$

  where the function F is simply an SVM classifier based on the component scores

HMM with Random Restarts

- A hidden Markov model can be viewed as a discrete hill climb technique

  As with any hill climb, when training an HMM we are only assured of a local maximum, and we can often significantly improve our results by executing the hill climb multiple times with different initial values, and selecting the best of the resulting models

- An HMM with random restarts can be seen as special case of **boosting**

  If we simply select the best model, then the "combining" function is particularly simple, and is given by:

$$F\big(\mathcal{S}(x;V,\Lambda_1),\mathcal{S}(x;V,\Lambda_2),\ldots,\mathcal{S}(x;V,\Lambda_\ell)\big) = \max\{\mathcal{S}(x;V,\Lambda_i)\}$$

HMM with Random Restarts

$$F\big(\mathcal{S}(x;V,\Lambda_1),\mathcal{S}(x;V,\Lambda_2),\ldots,\mathcal{S}(x;V,\Lambda_\ell)\big) = \max\{\mathcal{S}(x;V,\Lambda_i)\}$$

- Here, each scoring function is an HMM, where the trained models differ based only on different initial values

  Note that the "max" is the maximum over the HMM model scores, not the maximum over any particular set of input values

  ➢That is, we select the highest scoring model and use it for scoring

- Of course, we could use other combining functions, such as an average or majority vote of the corresponding classifiers

  In any case, since there is a score associated with each model generated by an HMM, any such combining function is well-defined

AdaBoost

- Given a collection of (weak) classifiers $c_1, c_2, \ldots, c_\ell$, AdaBoost is an iterative algorithm that generates a series of (generally, stronger) classifiers, $C_1, C_2, \ldots, C_M$ based on the classifiers c_i

Each classifier is determined from the previous classifier by the simple linear extension:

$$C_m(x) = C_{m-1}(x) + a_m c_i(x)$$

and the final classifier is given by $C = C_M$

- Note that at each iteration, we include a previously unused c_i from the set of (weak) classifiers and determine a new weight a_i. A **greedy approach** is used when selecting c_i, but it is **not a hill climb**, so that results might get worse at any step in the AdaBoost process

AdaBoost

- From this description, we see that AdaBoost fits the form

$$\widehat{\mathcal{S}}(x; V, \Lambda_i) = C_i(x)$$

and:

$$F\left(\widehat{\mathcal{S}}(x; V, \Lambda_1), \widehat{\mathcal{S}}(x; V, \Lambda_2), \ldots, \widehat{\mathcal{S}}(x; V, \Lambda_M)\right) = \widehat{\mathcal{S}}(x; V, \Lambda_M) = C_M(x)$$

AdaBoost

- AdaBoost is certainly the best-known boosting technique

- At each iteration, AdaBoost selects the “best” classifier from those available

  Where “best” is defined as the classifier that most improves on the overall accuracy of the new, combined classifier

  - That is, AdaBoost greedily selects a classifier that does the most to improve on the current iteration of the constructed classifier

- In AdaBoost the selected classifiers are combined as a weighted linear combination, where an optimal weight is calculated at each iteration, with all previously computed weights fixed