Ensamble Learning

ITAM

Outline

- Inspiration
  - Wisdom of the crowds
- Bagging
  - One instantiation: Random Forest
- Boosting
  - One instantiation: Adaboost

Inspiration

- Galton’s experiment
- Wisdom of crowds(Surowiecki)
- Who wants to be a millionaire

Ensamble

- These techniques construct multiple models
  - Diverse models, diverse aspects of the data
- They combine the prediction of several models
  - Voting, average, etc.
- The differences between techniques boils down to differences in these two points

Bagging

- Bagging (bootstrap aggregating)
- This method takes different random samples from the data (with replacement, boostrap)
- Here we assume that different samples will yield to important differences in the resulting models
- The final prediction is formed using some combination rule
  - Voting (mayority, plurality, weighed, ...)
  - Average (simple or weighed)

Bagging

- Since diversity is key, another idea is to foster it by using different subsets of the attributes for each model (subspace sampling)
  - Increases diversity
  - Increases speed
  - Helps a bit with the curse of dimensionality
- Decision trees are sensitive to variations in the attributes so they make good candidates for this ideas to work
  - Small variation can lead to trees with different constructions
- The above is the basis for a technique called Random Forests

Random Forest Training

- Input: Data D, number of trees T, number of attributes p
- Output: A set of trees
- for t=1 to T do
  - Create a sample D_t from D with replacement
  - Select p atributes at random and suppress the rest from D_t
  - Grow a decision tree A_t using D_t without prunning
- return $\{A_t \mid 1<=t<=T\}$
- Note : It is recommended that p=log(number of attributes in D) or p=sqrt(p=log(number of attributes in D)

Random Forest Prediction

- Input: Set of trees, an instance x to label
- Output: Prediction for x
- for t=1 to T do
  - y_t=At.predict(X)
- if classification
  - return vote({y_t|1<=t<=T})
- else
  - return mean({y_t|1<=t<=T})

Boosting

- This techniques comes from the question of whether the classes of problems weakly learnable are strongly learnable are equivalent
  - The answer was given by Schapire in the article “The Stength of Weak Learnability”
  - An the answer is Yes and the proof is by construction. The construction is called boosting (bootstrap aggreagation)
- General idea:
  - Generate a set of models sequentially. Each new model is trained to correct the errors of the previous models. The output is a combination of them all

AdaBoost

- Adaboost (adaptive boosting) is an implementation of this idea. In particular it establishes:
  - How to weigh the training examples to reflect the errors of the other models
  - How to weigh each model so as to reflect its importance and role in the final ensamble
- It uses an exponential cost function from which said weights are derived
- It takes as input any binary classification algorithm (assumes labels are -1 and 1)

Boosting

- New weights for the training examples are computed in each interation
  - This is implemented by sampling the training data with a new distribution

- The basic idea is to adjudicate half of the weight to the data that is correctly classified and the other half to the misclassified instances
  - Given the data D, each instance has an initial weight of 1/|D|
  - Subsequently given the classification error ε we assign half of the weight to the correctly classified instances and half to the misclassified ones
    - For example if we are wrong on 25% of the instances, the next model will assign these double their weight while the correcly classified will be reduced to 0.66

Boosting

Algorithm 11.3: Boosting($D, T, \mathscr{A}$) – train an ensemble of binary classifiers from reweighted training sets.

Input : data set D; ensemble size T; learning algorithm $\mathscr{A}$.
Output : weighted ensemble of models.

$w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$; // start with uniform weights
for $t = 1$ to T **do**
  run $\mathscr{A}$ on D with weights w_{ti} to produce a model M_t;
  calculate weighted error ϵ_t;
  if $\epsilon_t \geq 1/2$ **then**
    set $T \leftarrow t - 1$ and break
  end
  $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$; // confidence for this model
  $w_{(t+1)i} \leftarrow \frac{w_{ti}}{2\epsilon_t}$ for misclassified instances $x_i \in D$; // increase weight
  $w_{(t+1)j} \leftarrow \frac{w_{tj}}{2(1-\epsilon_t)}$ for correctly classified instances $x_j \in D$; // decrease weight
end
return $M(x) = \sum_{t=1}^{T} \alpha_t M_t(x)$

Algoritmo tomado del libro de Peter Flach (Machine Learning)

AdaBoost

- The choice of weights and alfa is related to minimizing the exponential cost function

- We wish to minimize

$$error = \sum_{i=1}^{N} e^{-m_i}$$

$$m_i = y_i \sum_{t=1}^{T} \alpha_t M_t(x_i)$$

Here y_i is the real class (1 o -1 y M_t is the predicted class)

Other methods

- The principle ideas of ensamble learning are:
  - Have different models that capture different patterns in the data
  - Have a way to combine them
- Having said this we could think of having an ensamble of methods using diverse techniques (neural networks + SVC + knn, etc.) with the idea to provide diversity
- We could think of combining them using yet another model, for instance a logistic regression
  - This is known as stacking
  - Now the mixing model has extra parameters to adjust (learn). You need to take this into account during the learning process

Exercise

- Download data from
  - http://archive.ics.uci.edu/ml/
    - I suggest Abalone
  - http://archive.ics.uci.edu/ml/datasets/Abalone?pagewanted=all
- Use Sklearn and compare a decision tree a random forest (adaboost is optional)