Meta-learning Methods-II

CS771: Introduction to Machine Learning

Purushottam Kar

Learning Ensembles

What are ensembles and why learn them?

- For sake of simplicity, focus only on binary classification only
- I wish to learn a classifier using a specific model class $\mathcal{M}$
- Why?
  - I have a nice implementation to learn from $\mathcal{M}$, or
  - Prediction using $\mathcal{M}$ is very cheap, or
  - Model class $\mathcal{M}$ is very powerful, or
  - Model class $\mathcal{M}$ is very popular, or
- But …
  - $\mathcal{M}$ has very high bias ☹ – what do I do?, or
  - $\mathcal{M}$ has very high variance ☹ - what do I do?
- Ensembles to the rescue!!

Voting Ensemble

- Receive K pre-trained classifiers $f_1, f_2, \ldots, f_K$ s.t. $f_i : \mathcal{X} \rightarrow \{-1, +1\}$
- Construct a new classifier $\hat{f}_{\text{MAJ}}$ such that for any $x \in \mathcal{X}$

$$\hat{f}_{\text{MAJ}}(x) = \text{sign}\left(\sum_{k=1}^{K} f_k(x)\right)$$

- Hope that mistakes of one will be corrected by others
- This is not expected to do well in general. If the classifiers were not trained properly, they may synchronize their mistakes
- Fixing this issue leads to useful techniques bagging and boosting
- Bagging applied to powerful models to reduce variance
- Boosting applied to weak models to reduce bias

Bagging – Bootstrap AGGregatING

- Given a training set S with n data points
  - Sample n data points with replacement from S call this S_1
  - Repeat this K times to obtain K bagged datasets $S_1, S_2, \ldots, S_K$
  - Learn a model $f_i: \mathcal{X} \rightarrow \{-1,1\}$ using dataset S_i (maybe the same algo)
  - Predict a new point $x \in \mathcal{X}$ using $\hat{f}_{\text{MAJ}}(x) = \text{sign}\left(\sum_{k=1}^{K} f_i(x)\right)$
- Can show that only about 63% of S lands up in any S_i
- This means all S_i have sufficient diff
- Even if we have a high variance method that overfits, it will overfit to substantially different sets
- Overall variance reduction effect

Bagging

- Note that the K models are trained independently
  - Allows for massive parallelization of learning algorithms
  - Can be a drawback too – more on this in a short while
- Can be seen as having regularization effect too
  - Works without the explicit use of regularizers
  - Useful in case where modifying code to add a regularizer is tricky
- Does not reduce bias because bagging usually applied to powerful models where bias is small to begin with but variance is high
- Two popular algorithms: random forests and dropout, have their genesis in bagging

Random Forests

- A collection of decision trees is called a decision forest
- Let us have training data $S = \{(\mathbf{x}^i, y^i)\}_{i=1,\ldots,n}$ with $x^i \in \mathbb{R}^d, y^i \in \{-1,1\}$
- Perform data bagging to get datasets $S_1, \ldots, S_K \subset S$
- Perform feature bagging to get $F_1, F_2, \ldots F_K \subset [d]$
  - Each F_i contains d' features sampled randomly **without** replacement
  - Typically $d' \sim \sqrt{d}$ for DTs
- Learn the k-th DT on dataset S_k using only the features in F_k
- Intuition: if some feature is really good, every tree will use it and then all trees will behave similarly so restrict available features

Dropout

- For a NN with N nodes, dropout attempts to train all 2^N possible subnetworks as the $K = 2^N$ models
- Intractable to execute explicitly which is why dropout does this approximately.
  - Parameter sharing: Two subnetworks that contain the same edge must have the same weight on that edge
  - Stochastic training: At every time step t a random subnetwork trained using a mini-batch of data (meant to approximate a bagged set)
- At test time dropout would like to have the effect of all 2^N models but so far no one has figured out how to do so tractably

Boosting

- Note that bagging does not explicitly force the K models to correct each others' errors – models are trained independently
- What if we ask model k to look at errors made by model $k-1$?
- Model k can then focus only on correcting errors of previous models
- This is basically the idea behind AdaBoost (ADAptive BOOSTing)
- Won Freund and Schapire the Godel prize
- Can train models which individually get no more than 51% accuracy and get an ensemble with as high as 99% accuracy*
- Can perform bias reduction by increasing the modelling power of the ensemble beyond the modelling power of any individual model
- Bagging could not claim any of this – only variance reduction

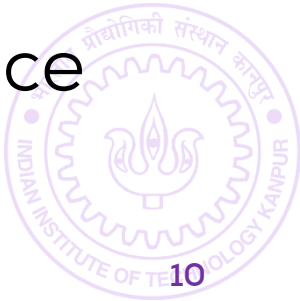
***some technical details missing**

AdaBoost

- AdaBoost has very simple requirements
- It requires a learning algorithm that can offer just greater than 51% classification accuracy for any *weighted* dataset given to it
- Dataset $S = \{(x^i, y^i, w^i)\}_{i=1,\ldots,n}$ with $x^i \in \mathcal{X}, y^i \in \{-1,1\}, w^i \in [0,1]$
- Adaboost assumes a learner $\mathcal{L}$ so that $\mathcal{L}(S) = f: \mathcal{X} \rightarrow \{-1,1\}$ s.t.

$$\sum_{i=1}^{n} w^i \cdot \mathbb{I}\{f(x^i) \neq y^i\} \leq 0.49$$

- Such learning algorithms are called *weak learners*
- Usually shallow decision trees or even linear classifiers suffice



AdaBoost

ADABOOST

1. Dataset $S = \{(x^i, y^i)\}_{i=1,\ldots,n}$, weak learner $\mathcal{L}$
2. Assign initial weights $w^{i,1} = \frac{1}{n}$ for all $i \in [n]$
3. For $t = 1,2, \ldots$
   1. Use $(S, \{w^{i,t}\})$ on $\mathcal{L}$ to learn $f_t: \mathcal{X} \to \{-1,1\}$
   2. Let $\epsilon_t \leftarrow \sum_{i=1}^{n} w^{i,t} \cdot \mathbb{I}\{f_t(x^i) \neq y^i\}$
   3. Let $\alpha_t \leftarrow \frac{1}{2} \cdot \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
   4. For each data point $i = 1,2, \ldots, n$
      1. Let $u^{i,t+1} \leftarrow w^{i,t} \cdot \exp\left(-\alpha_t y^i f_t(x^i)\right)$
   5. Let $w^{i,t+1} \leftarrow u^{i,t+1} / \sum_j u^{j,t+1}$
4. Repeat until convergence

AdaBoost

ADABOOST

1. Dataset $S = \{(x^i, y^i)\}_{i=1,\ldots,n}$, weak learner $\mathcal{L}$
2. Assign initial weights $w^{i,1} = \frac{1}{n}$ for all $i \in [n]$
3. For $t = 1,2, \ldots$
   1. Use $(S, \{w^{i,t}\})$ on $\mathcal{L}$ to learn $f_t: \mathcal{X} \to \{-1,1\}$
   2. Let $\epsilon_t \leftarrow \sum_{i=1}^{n} w^{i,t} \cdot \mathbb{I}\{f_t(x^i) \neq y^i\}$
   3. Let $\alpha_t \leftarrow \frac{1}{2} \cdot \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
   4. For each data point $i = 1,2, \ldots, n$
      1. Let $u^{i,t+1} \leftarrow w^{i,t} \cdot \exp\left(-\alpha_t y^i f_t(x^i)\right)$
   5. Let $w^{i,t+1} \leftarrow u^{i,t+1} / \sum_j u^{j,t+1}$
4. Repeat until convergence

Correctly classified points get downweighted. Misclassified points are upweighted

AdaBoost

ADABOOST

1. Dataset $S = \{(x^i, y^i)\}_{i=1,\ldots,n}$, weak learner $\mathcal{L}$
2. Assign initial weights $w^{i,1} = \frac{1}{n}$ for all $i \in [n]$
3. For $t = 1,2, \ldots$
   1. Use $(S, \{w^{i,t}\})$ on $\mathcal{L}$ to learn $f_t: \mathcal{X} \to \{-1,1\}$
   2. Let $\epsilon_t \leftarrow \sum_{i=1}^{n} w^{i,t} \cdot \mathbb{I}\{f_t(x^i) \neq y^i\}$
   3. Let $\alpha_t \leftarrow \frac{1}{2} \cdot \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
   4. For each data point $i = 1,2, \ldots, n$
      1. Let $u^{i,t+1} \leftarrow w^{i,t} \cdot \exp\left(-\alpha_t y^i f_t(x^i)\right)$
   5. Let $w^{i,t+1} \leftarrow u^{i,t+1} / \sum_j u^{j,t+1}$
4. Repeat until convergence

Correctly classified points get downweighted. Misclassified points are upweighted

Just to normalize the weights

AdaBoost

ADABOOST

1. Dataset $S = \{(x^i, y^i)\}_{i=1,\ldots,n}$, weak learner $\mathcal{L}$
2. Assign initial weights $w^{i,1} = \frac{1}{n}$ for all $i \in [n]$
3. For $t = 1,2, \ldots$
   1. Use $(S, \{w^{i,t}\})$ on $\mathcal{L}$ to learn $f_t: \mathcal{X} \to \{-1,1\}$
   2. Let $\epsilon_t \leftarrow \sum_{i=1}^{n} w^{i,t} \cdot \mathbb{I}\{f_t(x^i) \neq y^i\}$
   3. Let $\alpha_t \leftarrow \frac{1}{2} \cdot \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
   4. For each data point $i = 1,2, \ldots, n$
      1. Let $u^{i,t+1} \leftarrow w^{i,t} \cdot \exp\left(-\alpha_t y^i f_t(x^i)\right)$
   5. Let $w^{i,t+1} \leftarrow u^{i,t+1} / \sum_j u^{j,t+1}$
4. Repeat until convergence

Signifies the utility of f_t. As $\epsilon_t \to 0, \alpha_t \to \infty$

Correctly classified points get downweighted. Misclassified points are upweighted

Just to normalize the weights

AdaBoost

ADABOOST

1. Dataset $S = \{(x^i, y^i)\}_{i=1,\dots,n}$, weak learner $\mathcal{L}$
2. Assign initial weights $w^{i,1} = \frac{1}{n}$ for all $i \in [n]$
3. For $t = 1,2, \dots$
   1. Use $(S, \{w^{i,t}\})$ on $\mathcal{L}$ to learn $f_t: \mathcal{X} \to \{-1,1\}$
   2. Let $\epsilon_t \leftarrow \sum_{i=1}^{n} w^{i,t} \cdot \mathbb{I}\{f_t(x^i) \neq y^i\}$
   3. Let $\alpha_t \leftarrow \frac{1}{2} \cdot \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
   4. For each data point $i = 1,2, \dots, n$
      1. Let $u^{i,t+1} \leftarrow w^{i,t} \cdot \exp\left(-\alpha_t y^i f_t(x^i)\right)$
   5. Let $w^{i,t+1} \leftarrow u^{i,t+1} / \sum_j u^{j,t+1}$
4. Repeat until convergence

Final classifier used by AdaBoost is

$$\text{sign}\left(\sum_{t=1}^{T} \alpha_t f_t(x)\right)$$

Signifies the utility of f_t. As $\epsilon_t \to 0, \alpha_t \to \infty$

Correctly classified points get downweighted. Misclassified points are upweighted

Just to normalize the weights

AdaBoost

ADABOOST

1. $\{\ldots\}_{i=1,\ldots,n}$, weak learner $\mathcal{L}$
2. $w^{i,1} = \frac{1}{n}$ for all $i \in [n]$
3. For $t = 1,2, \ldots$
   1. Use $\left(S, \{w^{i,t}\}\right)$ on $\mathcal{L}$ to learn $f_t: \mathcal{X} \to \{-1,1\}$
   2. Let $\epsilon_t \leftarrow \sum_{i=1}^{n} w^{i,t} \cdot \mathbb{I}\{f_t(x^i) \neq y^i\}$
   3. Let $\alpha_t \leftarrow \frac{1}{2} \cdot \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
   4. For each data point $i = 1,2, \ldots, n$
      1. Let $u^{i,t+1} \leftarrow w^{i,t} \cdot \exp\left(-\alpha_t y^i f_t(x^i)\right)$
   5. Let $w^{i,t+1} \leftarrow u^{i,t+1} / \sum_j u^{j,t+1}$
4. Repeat until convergence

Usually all f_i are from the same model class $\mathcal{M}$ e.g. DTs

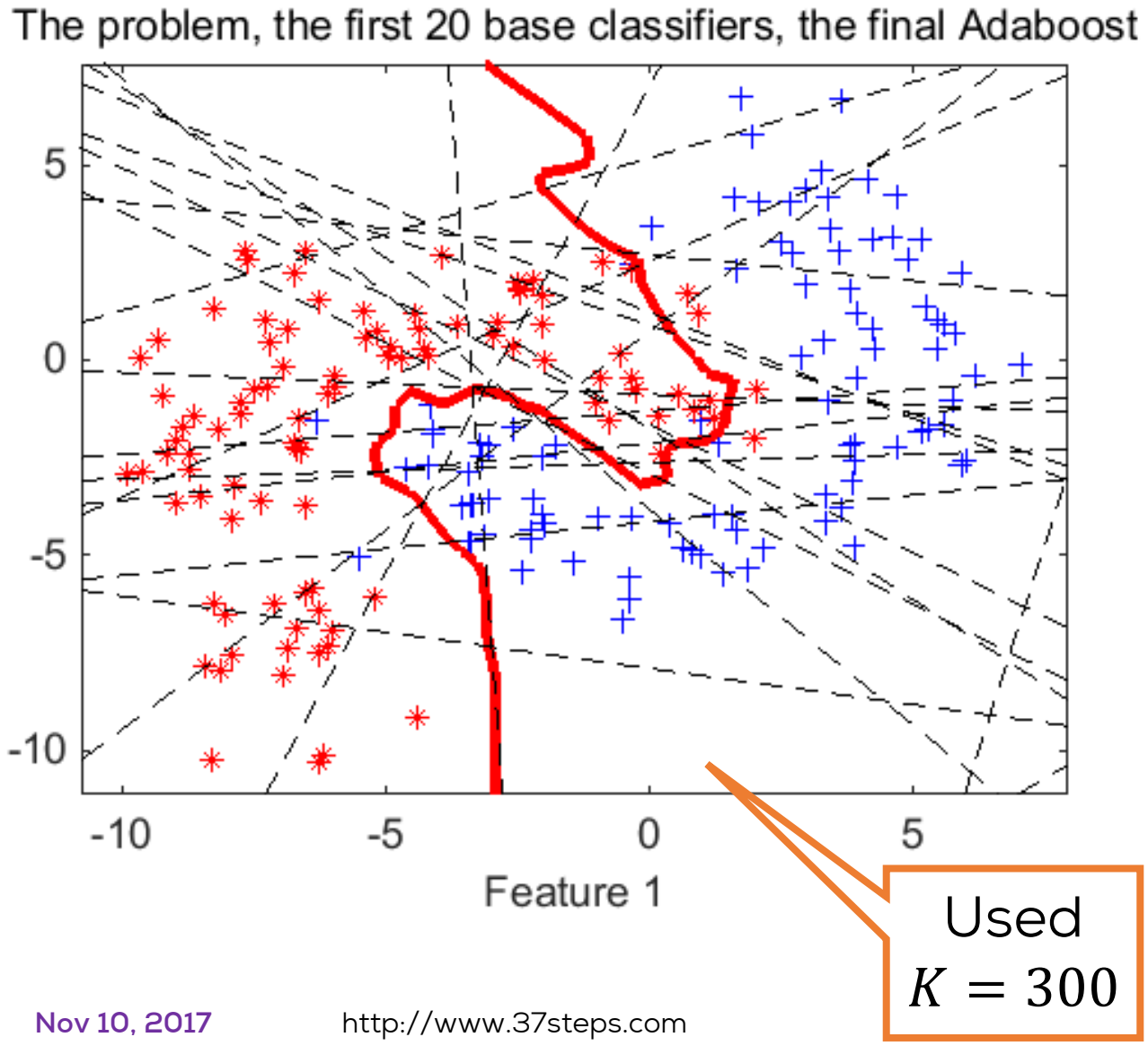
Final classifier used by AdaBoost is $\text{sign}\left(\sum_{t=1}^{T} \alpha_t f_t(x)\right)$

Signifies the utility of f_t. As $\epsilon_t \to 0, \alpha_t \to \infty$

Correctly classified points get downweighted. Misclassified points are upweighted

Just to normalize the weights

AdaBoost in action



- Linear classifiers as weak learners. Note that final classifier is

$$f(\mathbf{x}) = \text{sign}\left(\sum_{k=1}^{K} \alpha_k \cdot \text{sign}(\langle \mathbf{w}^k, \mathbf{x} \rangle)\right)$$

- Just like a 2 layer NN with sign as activation function!
- Note: if we had taken

$$f(\mathbf{x}) = \text{sign}\left(\sum_{k=1}^{K} \alpha_k \langle \mathbf{w}^k, \mathbf{x} \rangle\right)$$

would have been useless as it is just a linear classifier

A few thoughts

- Boosting is inherently sequential, cannot be parallelized easily
- Theoretical results show that AdaBoost seems to learn large-margin non-linear classifiers
- Also known that AdaBoost implicitly optimizes using the exponential loss $\ell(\hat{y}, y) = \exp(-y\hat{y})$
- Boosting cannot boost strong learners. No point applying AdaBoost with a 18 layer neural network
- Boosting also has variance reduction effects but not much since it is usually applied to simple models where variance is low anyway
- Variants of decision trees called "gradient boosted decision trees" some of the strongest generic classifiers around – perform boosting at level of gradient updates

Onto Recommendation Systems!

Lots of possibilities with ensembles. Most real-life challenges (NetFlix etc) are won by enormous ensembles, not single models

Please give your Feedback

http://tinyurl.com/ml17-18afb