Boosting

Tong Zhang

Rutgers University

Boosting

- Ensemble Learning algorithm.
- Given a learning algorithm $\mathcal{A}$:
  - how to generate the ensemble candidates?
  - how to combine the generated ensemble candidates?

Boosting

- Ensemble Learning algorithm.
- Given a learning algorithm $\mathcal{A}$:
  - how to generate the ensemble candidates?
  - how to combine the generated ensemble candidates?
- Invoke $\mathcal{A}$ with multiple samples (similar to Bagging).
  - goal: to find optimal ensemble by minimizing a loss function
  - learning method:
    - greedy, stage-wise optimization
    - invoking a base-learner (weak learner) $\mathcal{A}$.
    - adaptive resampling
- Bias reduction:
  - less stable but more expressive.
  - better than any single classifier.

Why Boosted Trees

- May build shallow trees
  - combine shallow trees (weak learner) to get strong learner.
- Linear model of high order features
  - automatically find high order interactive features $h_j(\cdot)$

$$h(x) = \sum_j w_j \underbrace{h_j(x)}_{\text{nonlinear in } x}$$

  - automatically handle heterogeneous features
  - high order features are indicator functions.
- Alternatives:
  - discretize each feature into (possibly overlapping) buckets
  - direct construction of feature combination.
  - nonlinear functions like kernels or neural networks.
  - direct greedy learning.

Weak Learning and Adaptive Resampling

- $\mathcal{A}$: a weak learner (e.g. shallow tree)
  - better than chance (0.5 error) on any (reweighted) training data.
- Question: can we combine weak learners to obtain a strong learner?
- Answer: yes, through adaptive resampling (boosting).
  - idea: overweighting difficult examples that are hard to classify.
- Compare with bagging: sampling without overweighting errors.
- Compare with outlier removal: underweighting errors.
  - reduce variance (but may increase bias)

The Idea of Adaptive Resampling

- Reweight the training data to overweight difficult examples.
- Using weak learner $\mathcal{A}$ to obtain classifiers f_j on reweighted samples.
- Adding the new classifier into ensemble, and choose weight w_j.
- Iterate.
- Final classifier is $\sum_j w_j f_j$.

AdaBoost (adaptive boosting)

- How to reweight, and how to compute w.
- Assume binary classification $y \in \{\pm 1\}$, and $f \in \{\pm 1\}$.

initialize sample weights $\{d_i\} = \{1/n\}$ for $\{(X_i, Y_i)\}$
for $j = 1, \cdots, J$
  call *Weak Learner* to obtain f_j using sample weighted by $\{d_i\}$
  let $r_j = \sum_i d_i f_j(X_i) Y_i$
  let $w_j = 0.5 \ln((1 + r_j)/(1 - r_j))$
  update d_i: $d_i \propto d_i e^{-w_j f_j(X_i) Y_i}$.
let $\bar{f}_J(x) = \sum_{j=1}^{J} w_j f_j(x)$

AdaBoost

Some Theoretical Results about AdaBoost

- Convergence: reduces margin error
  - f correctly classifies X_i with margin γ if $f(X_i)Y_i > \gamma > 0$.
  - If each weak learner f_j does better than $0.5 - \delta_j$ ($\delta_j > 0$) on reweighted samples with respect to classification error $I(f(X_i)Y_i \leq 0)$, then

$$\underbrace{\frac{1}{n}\sum_{i=1}^{n} I(\bar{f}_J(X_i)Y_i \leq \gamma)}_{\text{margin error}} \leq \exp(\gamma - 2\sum_{j=1}^{J}\delta_j^2).$$

- Generalization:
  - smaller margin error implies good generalization performance
- For linear separable problems, Adaboost does not usually maximize margin: different from SVM

Generalization Analysis for Boosting

- Generalization performance of $\hat{f} = \mathcal{A}(S_n)$: with probability at least $1 - \eta$,
$$\text{test error} \leq \text{training error} + \text{model complexity}.$$
- Decision tree of fixed depth: $\mathcal{H}$ has finite VC-dimension d_{VC}, $(\phi(f, y) = I(fy \leq 0))$:
$$\text{test error} \leq \text{training error} + C\sqrt{\frac{1}{n}(d_{VC} - \ln(\eta))}$$
$$\text{test error} \leq 2 \times \text{training error} + \frac{C}{n}(d_{VC} - \ln(\eta)).$$
- Traditional analysis without considering margin

Generalization Error using Number of Steps

- $\mathcal{H}$: VC-dimension d_{VC}.
- Ensemble $\bar{f}_J = \sum_{i=1}^{J} w_i f_i(x) : f_i \in \mathcal{H}\}$:

$$\underbrace{R(\bar{f}_J)}_{\text{test error}} \leq 2 \underbrace{\hat{R}(\bar{f}_J)}_{\text{training error}} + \underbrace{\frac{C}{n}(Jd_{VC} - \ln(\eta))}_{\text{complexity linear in } J}.$$

- $\bar{f}_J$: boosted tree after J round:
  - training error: $O(e^{-2J\delta^2})$ ($0.5 - \delta$ error reduction)
  - generalization error

$$R(\bar{f}_J) \leq O(e^{-J\gamma}) + \frac{C}{n}(Jd_{VC} - \ln(\eta)).$$

Generalization Error Anomaly

- Empirical observations:
  - AdaBoost is difficult to overfit.
  - even when training error becomes zero, generalization error still decays
- Not explained by the generalization bound using the number of steps.
- require additional analysis: margin

Margin Bound

- Decision tree of fixed depth: $\mathcal{H}$ has finite VC-dimension d_{VC}, then

$$\text{training error} \leq 2 \times \text{margin error} + \text{fixed complexity}$$

$$\mathbf{E}_{X,Y} I(\bar{f}_J(X)Y \leq 0) \leq \underbrace{\frac{2}{n}\sum_{i=1}^{n} I(\hat{f}_m(X_i)Y_i \leq \gamma\|w\|_1)}_{\to 0 \text{ when } J \to \infty} + \underbrace{\frac{C}{n}\left(\frac{d_{VC}}{\gamma^2} - \ln(\eta)\right)}_{\text{independent of } J}.$$

- Explains why AdaBoost can keep improving even when classification error becomes zero
  - reason: margin error decreases

Margin Analysis and L_1 Regularization

- Margin analysis is a special case of general L_1 regularization
- Let ϕ be a smooth loss.
- Given L_1 constraint $\sum_j w_j \leq A$:

$$\mathbf{E}_{X,Y}\phi(\bar{f}_J(X), Y) \leq \frac{1}{n}\sum_{i=1}^{n}\phi(\bar{f}_J(X_i), Y_i) + C_\phi\sqrt{\frac{1}{n}(A^2 d_{VC} - \ln(\eta))}.$$

Complexity measured by A, not number of steps J.

Summary of Generalization Analysis

- Estimate generalization of boosting: using the following complexity control
  - L_1: 1-norm of the weights w_j are bounded.
  - L_0: number of boosting steps (sparse representation).
- Which complexity control is better?
  - sparsity is more fundamental but both views are useful.
  - can be more refined analysis in between.
- In more general boosting methods:
  - complexity can be controlled either by L_1 (1-norm) or L_0 (sparsity).

Issues corresponding to the Weak Learner View

- Weak learner: this is only an assumption, how to prove existence?
  - what is a weak learner?
  - why boosted tree works, and boosted SVM does not.
- Overfitting: driving error to zero can overfit the data (for non-separable problems)
- AdaBoost does not maximize margin.
- Adaptive resampling: why this specific form.
- Can we generalize adaptive resampling idea to regression and complex loss functions?

From Adaptive Resampling to Greedy Boosting

- Weak learner: picks f_j from a hypothesis space $\mathcal{H}_j$ to minimize certain error criterion.
- Goal: find $w_j \geq 0$ and $f_j \in \mathcal{H}_j$ to minimize loss

$$[\{\hat{w}_j, \hat{f}_j\}] = \arg \min_{\{w_j \geq 0, f_j \in \mathcal{H}_j\}} \sum_{i=1}^{n} \phi\left(\sum_j w_j f_j(X_i), Y_i\right). \qquad (*)$$

- Idea: greedy optimization.
  - at stage j: fix (w_k, f_k) $(k < j)$, find (w_j, f_j) to minimize the loss $(*)$.

AdaBoost as Greedy Boosting

- Loss $\phi(f, y) = \exp(-fy)$.
- Goal: using greedy boosting to minimize

$$[\{\hat{w}_j, \hat{f}_j\}] = \arg \min_{\{w_j \geq 0, f_j \in \mathcal{H}_j\}} \sum_{i=1}^{n} e^{-\sum_j w_j f_j(X_i) Y_i}.$$

- Greedy optimization: at stage j, let $d_i \propto e^{-\sum_{k=1}^{j-1} \hat{w}_k \hat{f}_k(X_i) Y_i}$, and solve

$$[\hat{w}_j, \hat{f}_j] = \arg \min_{w_j \geq 0, f_j \in \mathcal{H}_j} \sum_{i=1}^{n} d_i e^{-w_j f_j(X_i) Y_i}.$$

- It can be shown solution is exactly the Adaboost update.

General Loss Function

- Learn prediction function h(x).
- By solving learning formulation

$$\hat{h} = \arg\min_{h \in H} \mathcal{L}(h)$$

  - $\mathcal{L}(h)$: complex loss function of the form

$$\mathcal{L}(h) = \frac{1}{n} \sum_{i=1}^{n} \phi_i(h(x_{i,1}), \cdots, h(x_{i,m_i}), y_i)$$

- Greedy algorithm: generalization of Adaboost
  - $(s_k, g_k) = \arg\min_{g \in C, s \in R} \mathcal{L}(h_k + sg)$
  - $h_{k+1} \leftarrow h_k + \tilde{s}_k g_k$ ($\tilde{s}_k$ may not equal s_k)

General Loss Function

- Learn prediction function h(x).
- By solving learning formulation

$$\hat{h} = \arg\min_{h \in H} \mathcal{L}(h)$$

  - $\mathcal{L}(h)$: complex loss function of the form

$$\mathcal{L}(h) = \frac{1}{n} \sum_{i=1}^{n} \phi_i(h(x_{i,1}), \cdots, h(x_{i,m_i}), y_i)$$

- Greedy algorithm: generalization of Adaboost
  - $(s_k, g_k) = \arg\min_{g \in C, s \in R} \mathcal{L}(h_k + sg)$
  - $h_{k+1} \leftarrow h_k + \tilde{s}_k g_k$ ($\tilde{s}_k$ may not equal s_k)
- However, this greedy weak learner is specialized and hard to implement; can we simplify?

Boosting with Regression base Learner

- Simplified weak learner: nonlinear regression base leaner $\mathcal{A}$.
  - input: $X = [x_1, \ldots, x_k]$, residues $R = [r_1, \ldots, r_k]$
  - output: a nonlinear function $\hat{g} = \mathcal{A}(X, R) \in \mathcal{C}$ (e.g. decision tree)

$$\sum_{j=1}^{k} (\hat{g}(x_j) - r_j)^2 \approx \min_{g \in \mathcal{C}} \sum_{j=1}^{k} (g(x_j) - r_j)^2.$$

Boosting with Regression base Learner

- Simplified weak learner: nonlinear regression base leaner $\mathcal{A}$.
  - input: $X = [x_1, \ldots, x_k]$, residues $R = [r_1, \ldots, r_k]$
  - output: a nonlinear function $\hat{g} = \mathcal{A}(X, R) \in \mathcal{C}$ (e.g. decision tree)

$$\sum_{j=1}^{k} (\hat{g}(x_j) - r_j)^2 \approx \min_{g \in \mathcal{C}} \sum_{j=1}^{k} (g(x_j) - r_j)^2.$$

- Question: can we use $\mathcal{A}$ to optimize complex loss functions $\mathcal{L}(\cdot)$?
- Answer: yes:
  - functional gradient boosting (Friedman 01)
  - based on a functional generalization of gradient descent
  - a generalization of Adaboost

Gradient Boosting Algorithm

$h_0(x) = 0$
for $t = 1$ to T **do**
  $r_t = \partial \mathcal{L}(h, Y)/\partial h|_{h=h_{t-1}(X)}$
  $g_t = \mathcal{A}(X, r_t)$
    // (i.e. call base learner) $g_t \approx \arg\min_{g \in \mathcal{C}} \|g(X) - r_t\|_2^2$
  $\beta_t = \arg\min_\beta \mathcal{L}(h_{t-1}(X) + \beta \cdot g_t(X), Y)$
  $h_t(x) = h_{t-1}(x) + s_t \cdot \beta_t g_t(x)$
end for
Return $h_T(x)$

- $s_t = s$: shrinkage parameter — convergence requires $s \approx 0$
- functional generalization of gradient descent
  $h_t \leftarrow h_{t-1} - s_t \partial \mathcal{L}(h_t)/\partial h_t$

Why Boosted Trees

- Linear model of high order features
- Automatically handle heterogeneous features
  - create new (high order) features that are indicator functions.
- Automatically find high order interactive features
  - through tree splitting procedure.
  - a method to solve the problem of huge search space.
    - assume good high order features depend on actively maintained set of (good) features constructed so far.
- Alternatives:
  - discretize each feature into (possibly overlapping) buckets
  - direct construction of feature combination.
  - nonlinear functions like kernels or neural networks.
  - nonlinear feature learning using coding
  - general greedy feature learning by maintaining a set of features and adding new ones.