Ensembles: Random Forests and Boosting

Lyle Ungar

Learning objectives
Ensembles: random forests
Review stagewise regression
Know adaboost well
See gradient tree boosting

Ensemble: average many predictors

- **Ensemble method**
  - Weighted combination of T weak models: $h_t(\boldsymbol{x})$

$$h(\mathbf{x}) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$$

  - Often $\alpha_t = 1$
- **For real values, average $h_t(x)$**
  - *I.e., instead of taking the sign, divide by* $\sum_{t=1}^{T} \alpha_t$

Ensembles are great!!!

- Why?

Bagging

- **Generate $h_t(x)$ by resampling a fraction f of the n training points for each of T training sets**

$$h(\mathbf{x}) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$$

  - Often $\alpha_t = 1$

- **For real values, often average $h_t(x)$**

When is bagging a good idea?

- Linear regression?
- Decision trees?
- Deep learning?

When is bagging a good idea?

- **Linear regression?**
  - **No;** when you add a bunch of linear regressions, you still get a linear regression
- **Decision trees?**
  - **Yes;** when you add a bunch of decision trees you get a much more complex decision surface.
- **Deep learning?**
  - It gives better accuracy, but mostly people don't do it because it is too expensive

Random Forests

- **Repeat k times:**
  - Choose a training set by choosing $f{\cdot}n$ training cases
    - with replacement ('bootstrapping')
  - Build a decision tree as follows
    - For each node of the tree, randomly choose *m* features and find the best split from among them
  - Repeat until the tree is built
- ***To predict, take the modal prediction of the k trees***

Typical values:
k* = *1,000*   *m* = *sqrt(p)

Random forests are widely used

- **They don't overfit**
  - Why not?
  - Where is the regularization?
- **They don't underfit (much)**
  - Why are they so much better than decision trees?
  - Than logistic regression?

Questions?

Top

Stagewise Regression

- **Sequentially learn the weights α_t**
  - Never readjust previously learned weights

$$h(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t \, \phi_t(\mathbf{x})$$

$h_0(\mathbf{x}) = 0$

For $t = 1{:}T$

$r_t = y - h_{t-1}(\mathbf{x})$ *find residual*

pick $\phi_t(\mathbf{x})$ *pick next feature*

regress $r_t = \alpha_t \, \phi_t(\mathbf{x})$ **to find** α_t

$h_t(\mathbf{x}) = h_{t-1}(\mathbf{x}) + \alpha_t \, \phi_t(\mathbf{x})$ *update model*

Boosting

- **Ensemble method**
  - Weighted combination of weak learners $h_t(\boldsymbol{x})$

$$h(\mathbf{x}) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$$

- **Estimated stagewise**
  - At each stage, boosting gives more weight to what it got wrong before

Adaboost

Given: n examples $(\mathbf{x}_i, y_i)$, where $\mathbf{x} \in \mathcal{X}, y \in \pm 1$.

Initialize: $D_1(i) = \frac{1}{n}$

For $t = 1 \ldots T$

- Train weak classifier on distribution $D(i)$, $h_t(\mathbf{x}) : \mathcal{X} \mapsto \pm 1$
- Choose weight α_t (see how below)
- Update: $D_{t+1}(i) = \frac{D_t(i) \exp\{-\alpha_t y_i h_t(\mathbf{x}_i)\}}{Z_t}$, for all i, where $Z_t = \sum_i D_t(i) \exp\{-\alpha_t y_i h_t(\mathbf{x}_i)\}$

Output classifier: $h(\mathbf{x}) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$

Where α_t is the log-odds of the weighted probability of the prediction being wrong

$$\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t} \qquad \epsilon_t = \sum_i D_t(i) \mathbf{1}(y_i \neq h_t(\mathbf{x_i}))$$

Adaboost example

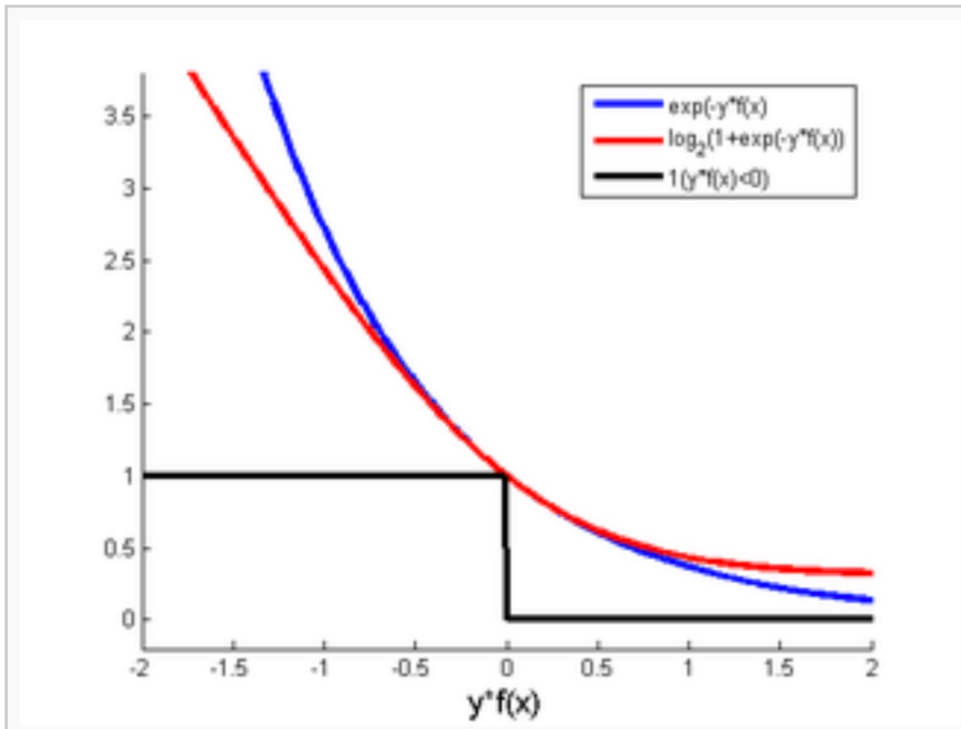
- https://alliance.seas.upenn.edu/~cis520/dynamic/2021/wiki/index.php?n=Lectures.Boosting

Questions?

Top

Adaboost minimizes exponential loss

$$\textbf{Boosting}: \quad \exp(-y_i f_\alpha(\mathbf{x}_i)) \quad \textbf{Logistic}: \quad \log(1 + \exp(-y_i f_{\mathbf{w}}(\mathbf{x}_i)))$$



And it learns it exponentially fast

$$\frac{1}{n}\sum_i \mathbf{1}(y_i \neq h(\mathbf{x}_i)) \leq \prod_{t=1}^{T} Z_t \leq \exp\{\sum_t -2(0.5-\epsilon_t)^2\} \leq \exp\{-2T\gamma^2\}$$

Average Error

Exponential in stages T and the accuracy of the weak learner γ

where $\gamma = \min_t(0.5 - \epsilon_t)$.

Gradient Tree Boosting

- **Current state-of-the-art for moderate-sized data sets**
  - on average very slightly better than random forests
- **Ensemble of Trees**
  - Adaboost used ‘stumps’

Gradient Boosting

- **Model:** $h(\mathbf{x}) = \Sigma_t \, \alpha_t \, h_t(\mathbf{x}) + const$
- **Loss function:** $L(\mathbf{y}, h(\mathbf{x}))$
  - L_2 or logistic or …
- **Base learner:** $h_t(\mathbf{x})$
  - Decision tree of specified depth
- **Optionally subsample features**
  - "stochastic gradient boosting"
- **Do stagewise estimation of h(x)**
  - Estimate $h_t(x)$ *and* α_t at each iteration t

Gradient Tree Boosting for Regression

- **Loss function: L_2**
- **Base learners** $h_t(\mathbf{x})$
  - Fixed-depth regression tree fit on residual
  - Gives a constant prediction for each leaf of the tree
- **Stagewise: find weights on each** $h_t(\mathbf{x})$
  - Fancy version: fit different weights for each leaf of tree

Gradient Tree Boosting

- **Stagewise estimation** $\quad h(x) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$
- *For L_2 loss*

Initialize $h_0(\mathbf{x})$ = average y

For $t = 1{:}T$

pick fraction f **of** n **observations** — *bag*

$r_t = y - h_{t-1}(\mathbf{x})$ — *find residual*

fit decision tree: $\phi_t(\mathbf{x})$ — *pick weak learner*

regress $r_t = \alpha_t\, \phi_t(\mathbf{x})$ **to find** α_t — *not needed here*

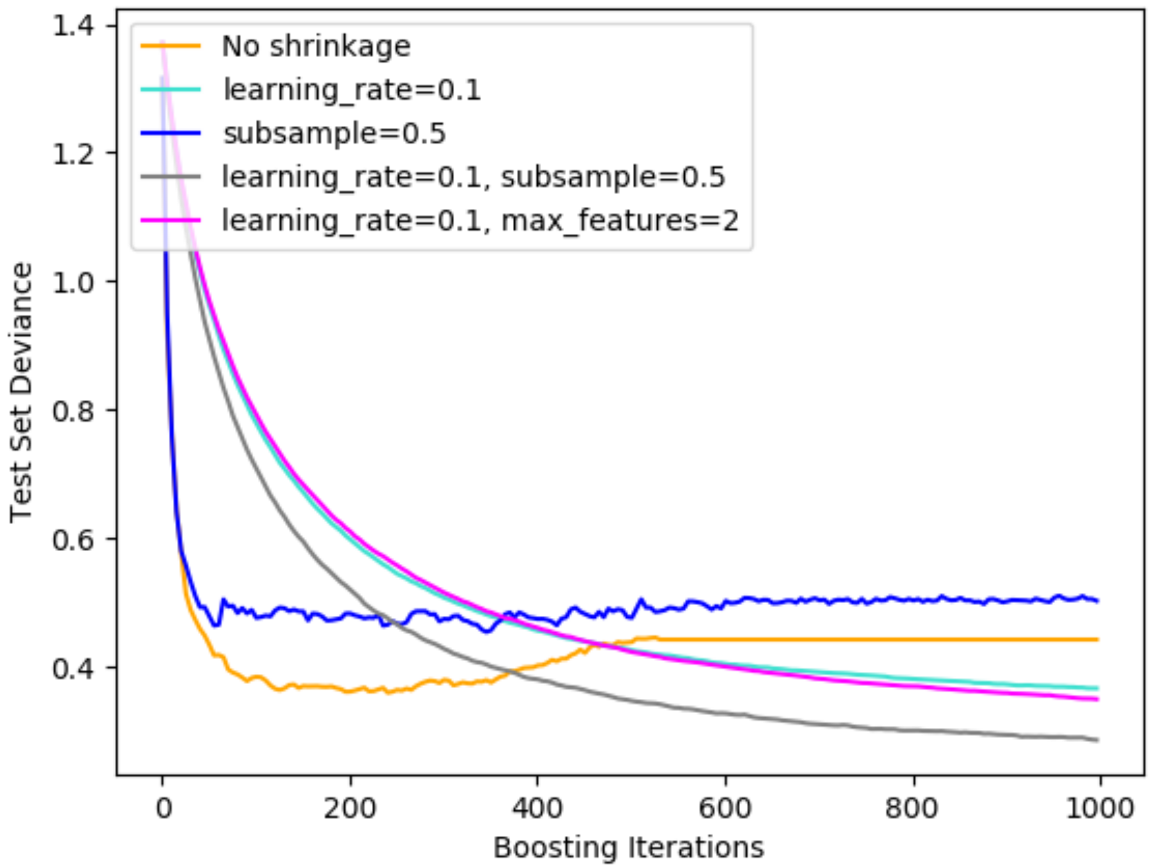
$h_t(\mathbf{x}) = h_{t-1}(\mathbf{x}) + \eta\, \alpha_t\, \phi_t(\mathbf{x})$ — *update model*

learning rate η

Gradient Tree Boosting Regularization

- **Tree depth:** d
- **Number of stages:** T
- **Bag size:** $f\,n$
- **Learning rate:** η

Regularization helps



Subsample = stochastic gradient boosting

Learning rate = shrinkage on α

http://scikit-learn.org/stable/auto_examples/ensemble/plot_gradient_boosting_regularization.html

What you should know

- **Boosting**
  - Stagewise regression, upweighting previous errors
  - Gives highly accurate ensemble models
  - Relatively fast
  - Tends not to overfit (but still: use early stopping!)
- **Gradient Tree Boosting**
  - ”base learner” is a decision tree
  - Stagewise (on pseudo-residuals)
  - Very accurate!!!

Questions?

Top