Ensembles: Random Forests and Boosting

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

Lyle Ungar
Ensemble: average many predictors

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

  \[
  h(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)
  \]

  - 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!!!
Bagging

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

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

- 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:
\[
k = 1,000 \quad m = \sqrt{p}
\]
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?
Stagewise Regression

- Sequentially learn the weights $\alpha_t$
  - Never readjust previously learned weights

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

$h_0(x) = 0$

For $t = 1:T$

- $r_t = y - h_{t-1}(x)$ find residual
- Pick $\phi_t(x)$ pick next feature
- Regress $r_t = \alpha_t \phi_t(x)$ to find $\alpha_t$
- $h_t(x) = h_{t-1}(x) + \alpha_t \phi_t(x)$ update model
Boosting

- **Ensemble method**
  - Weighted combination of weak learners $h_t(x)$

\[ h(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right) \]

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

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

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

For \(t = 1 \ldots T\)

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

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

Where \(\alpha_t\) is the log-odds of the weighted probability of the prediction being wrong

\[
\alpha_t = \frac{1}{2} \log \frac{1-e_t}{e_t} \quad \epsilon_t = \sum_i D_t(i) 1(y_i \neq h_t(x_i))
\]
Adaboost example

Questions?
Adaboost minimizes exponential loss

Boosting: \( \exp(-y_if_\alpha(x_i)) \)  Logistic: \( \log(1 + \exp(-y_if_w(x_i))) \)
And it learns it exponentially fast

\[ \frac{1}{n} \sum_i 1(y_i \neq h(x_i)) \leq \prod_{t=1}^T Z_t \leq \exp\left\{ \sum_t -2(0.5 - \epsilon_t)^2 \right\} \leq \exp\{-2T\gamma^2\} \]

Average Error

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

Exponential in stages \( T \) and the accuracy of the weak learner \( \gamma \).
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**: \( h(x) = \sum_t \alpha_t h_t(x) + \text{const} \)
- **Loss function**: \( L(y, h(x)) \)
  - \( L_2 \) or logistic or …
- **Base learner**: \( h_t(x) \)
  - Decision tree of specified depth
- **Optionally subsample features**
  - “stochastic gradient boosting”
- **Do stagewise estimation of \( h(x) \)**
  - Estimate \( h_t(x) \) and \( \alpha_t \) at each iteration \( t \)
1. Initialize model with a constant value:

\[ F_0(x) = \arg \min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma). \]

2. For \( m = 1 \) to \( M \):

   1. Compute so-called pseudo-residuals:

   \[
   r_{im} = -\left[ \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x) = F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.
   \]

   2. Fit a base learner (e.g. tree) \( h_m(x) \) to pseudo-residuals, i.e. train it using the training set \( \{(x_i, r_{im})\}_{i=1}^{n} \).

   3. Compute multiplier \( \gamma_m \) by solving the following one-dimensional optimization problem:

   \[
   \gamma_m = \arg \min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)).
   \]

   4. Update the model:

   \[
   F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).
   \]

3. Output \( F_M(x) \).
Gradient Tree Boosting for Regression

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

Subsample = stochastic gradient boosting

Learning rate = shrinkage on $\alpha$

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?