Learning objectives
Review stagewise regression
Know adaboost and gradient boosting algorithms

Boosting

Lyle Ungar
Stagewise Regression

- Sequentially learn the weights $\alpha_t$
  - Never readjust previously learned weights
  - $h(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$

$h(x) = 0$ or average($y$)

For $t = 1:T$

- $r_t = y - h(x)$
- regress $r_t = \alpha_t h(x)$ to find $\alpha_t$
- $h(x) = h(x) + \alpha_t h_t(x)$
Boosting

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

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

◆ Learned 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

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

Adaboost minimizes exponential loss

Boosting: \( \exp(-y_if(x_i)) \)

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

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

**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**
  - i.e. on average very slightly better than random forests when you don’t have enough data to do deep learning
Gradient Boosting

- **Model:** $F(x) = \sum_i \gamma_i h_i(x) + \text{const}$
- **Loss function:** $L(y, F(x))$
  - $L_2$ or logistic or ...
- **Base learner:** $h_i(x)$
  - Decision tree of specified depth
- **Optionally subsample features**
  - "stochastic gradient boosting"
- **Do stagewise estimation of $F(x)$**
  - Estimate $h_i(x)$ and $\gamma_i$ at each iteration $i$
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**:

      $$r_{im} = -\left[ \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x) = F_{m-1}(x)}$$

      for $i = 1, \ldots, n$.  

      For squared error, this is just the standard residual

   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** $h_i(x)$
  - Fixed depth regression tree fit on pseudo-residual
  - Gives a constant prediction for each leaf of the tree
- **Stagewise:** find weights on each $h_i(x)$
  - Fancy version: fit different weights for each leaf of tree
Regularization helps

Subsample = stochastic gradient boosting

Learning rate = shrinkage on $\gamma$

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**
  - Uses pseudo-residuals
  - Very accurate!!!