Trees

Machine Learning – CSE546
Kevin Jamieson
University of Washington

October 26, 2017
Trees

Build a binary tree, splitting along axes

\[ f(x) = \sum_{m=1}^{M} c_m I(x \in R_m). \]
Trees

Build a binary tree, splitting along axes

\[ f(x) = \sum_{m=1}^{M} c_m I(x \in R_m). \]

How do you split?

When do you stop?
Learning decision trees

- Start from empty decision tree
- Split on **next best attribute (feature)**
  - Use, for example, information gain to select attribute
  - Split on
- Recurse
- Prune

\[
\begin{align*}
\arg \max_i IG(X_i) &= \arg \max_i H(Y) - H(Y \mid X_i) \\

f(x) &= \sum_{m=1}^M c_m I(x \in R_m).
\end{align*}
\]
Trees

- Trees
  - have low bias, high variance
  - deal with categorical variables well
  - intuitive, interpretable
  - good software exists
  - Some theoretical guarantees

\[ f(x) = \sum_{m=1}^{M} c_m I(x \in R_m). \]
Random Forests

Machine Learning – CSE546
Kevin Jamieson
University of Washington

October 26, 2017
Random Forests

Tree methods have **low bias** but **high variance**.

One way to reduce variance is to construct a lot of “lightly correlated” trees and average them:

“Bagging:” Bootstrap aggregating
Random Forrests

Algorithm 15.1 Random Forest for Regression or Classification.

1. For $b = 1$ to $B$:
   
   (a) Draw a bootstrap sample $Z^*$ of size $N$ from the training data.
   
   (b) Grow a random-forest tree $T_b$ to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size $n_{min}$ is reached.
      
      i. Select $m$ variables at random from the $p$ variables.
      
      ii. Pick the best variable/split-point among the $m$.
      
      iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point $x$:

Regression: $\hat{f}_{rf}^B(x) = \frac{1}{B} \sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the $b$th random-forest tree. Then $\hat{C}_{rf}^B(x) = \text{majority vote} \{\hat{C}_b(x)\}_1^B$.

$m \sim \text{sqrt}(p), p/3$
The Kinect pose estimation pipeline

capture depth image & remove bg

infer body parts per pixel

cluster pixels to hypothesize body joint positions

fit model & track skeleton

Random Forrest

Random forest

3 nearest neighbor
Given random variables $Y_1, Y_2, \ldots, Y_B$ with
\[ \mathbb{E}[Y_i] = y, \mathbb{E}[(Y_i - y)^2] = \sigma^2, \mathbb{E}[(Y_i - y)(Y_j - y)] = \rho \sigma^2 \]

The $Y_i$'s are identically distributed but not independent.

\[
\mathbb{E}\left(\frac{1}{B} \sum_{i=1}^{B} (Y_i - y)^2\right) = \mathbb{E}\left[\frac{1}{B^2} \left( \sum_{i=1}^{B} (Y_i - y)^2 + \sum_{i \neq j} (Y_i - y)(Y_j - y) \right) \right]
\]

\[
= \frac{1}{B^2} \left( \sum_{i=1}^{B} \sigma^2 + \sum_{i \neq j} \rho \sigma^2 \right)
\]

\[
= \frac{1}{B} \sigma^2 + \frac{\beta(\beta - 1)}{B^2} \rho \sigma^2
\]

\[
= \frac{\sigma^2}{B} + \rho \sigma^2
\]
Random Forests

- Random Forests
  - have low bias, low variance
  - deal with categorical variables well
  - not that intuitive or interpretable
  - good software exists
  - Some theoretical guarantees
  - Can still overfit
Boosting

Machine Learning – CSE546
Kevin Jamieson
University of Washington

October 26, 2017
Boosting

• 1988 Kearns and Valiant: “Can weak learners be combined to create a strong learner?”

Weak learner definition (informal):

An algorithm $A$ is a weak learner for a hypothesis class $\mathcal{H}$ that maps $\mathcal{X}$ to $\{-1, 1\}$ if for all input distributions over $\mathcal{X}$ and $h \in \mathcal{H}$, we have that $A$ correctly classifies $h$ with error at most $1/2 - \gamma$.
Boosting

• 1988 Kearns and Valiant: “Can weak learners be combined to create a strong learner?”

Weak learner definition (informal):

An algorithm $A$ is a weak learner for a hypothesis class $\mathcal{H}$ that maps $\mathcal{X}$ to $\{-1, 1\}$ if for all input distributions over $\mathcal{X}$ and $h \in \mathcal{H}$, we have that $A$ correctly classifies $h$ with error at most $1/2 - \gamma$

• 1990 Robert Schapire: “Yup!”

• 1995 Schapire and Freund: “Yes, practically” AdaBoost
Boosting

• 1988 Kearns and Valiant: “Can weak learners be combined to create a strong learner?”

Weak learner definition (informal):

An algorithm $A$ is a weak learner for a hypothesis class $\mathcal{H}$ that maps $\mathcal{X}$ to $\{-1, 1\}$ if for all input distributions over $\mathcal{X}$ and $h \in \mathcal{H}$, we have that $A$ correctly classifies $h$ with error at most $1/2 - \gamma$

• 1990 Robert Schapire: “Yup!”
• 1995 Schapire and Freund: “Yes, practically” AdaBoost
• 2014 Tianqi Chen: “Scale it up!” XGBoost
Boosting and Additive Models

Machine Learning – CSE546
Kevin Jamieson
University of Washington

October 26, 2017
Additive models

- Consider the first algorithm we used to get good classification for MNIST. Given: \( \{(x_i, y_i)\}_{i=1}^{n} \) \( x_i \in \mathbb{R}^d, y_i \in \{-1, 1\} \)
- Generate **random** functions: \( \phi_t : \mathbb{R}^d \to \mathbb{R} \quad t = 1, \ldots, p \)
- Learn some weights: \( \hat{w} = \arg \min_w \sum_{i=1}^{n} \text{Loss} \left( y_i, \sum_{t=1}^{p} w_t \phi_t(x_i) \right) \)
- Classify new data: \( f(x) = \text{sign} \left( \sum_{t=1}^{p} \hat{w}_t \phi_t(x) \right) \)
Additive models

• Consider the first algorithm we used to get good classification for MNIST. Given: \( \{(x_i, y_i)\}_{i=1}^{n} \) \( x_i \in \mathbb{R}^d, y_i \in \{-1, 1\} \)

• Generate **random** functions: \( \phi_t : \mathbb{R}^d \rightarrow \mathbb{R} \quad t = 1, \ldots, p \)

• Learn some weights: \( \hat{w} = \arg \min_w \sum_{i=1}^{n} \text{Loss} \left( y_i, \sum_{t=1}^{p} w_t \phi_t(x_i) \right) \)

• Classify new data: \( f(x) = \text{sign} \left( \sum_{t=1}^{p} \hat{w}_t \phi_t(x) \right) \)

An interpretation:
Each \( \phi_t(x) \) is a classification rule that we are assigning some weight \( \hat{w}_t \)
Additive models

- Consider the first algorithm we used to get good classification for MNIST. Given: \( \{(x_i, y_i)\}_{i=1}^n \) \( x_i \in \mathbb{R}^d, y_i \in \{-1, 1\} \)
- Generate random functions: \( \phi_t : \mathbb{R}^d \to \mathbb{R} \quad t = 1, \ldots, p \)
- Learn some weights: \( \hat{w} = \arg\min_w \sum_{i=1}^n \text{Loss} \left( y_i, \sum_{t=1}^p w_t \phi_t(x_i) \right) \)
- Classify new data: \( f(x) = \text{sign} \left( \sum_{t=1}^p \hat{w}_t \phi_t(x) \right) \)

An interpretation:
Each \( \phi_t(x) \) is a classification rule that we are assigning some weight \( \hat{w}_t \)

\[
\hat{w}, \hat{\phi}_1, \ldots, \hat{\phi}_t = \arg\min_{w, \phi_1, \ldots, \phi_p} \sum_{i=1}^n \text{Loss} \left( y_i, \sum_{t=1}^p w_t \phi_t(x_i) \right)
\]

is in general computationally hard
**Forward Stagewise Additive models**

\[ b(x, \gamma) \] is a function with parameters \( \gamma \)

**Examples:**

\[ b(x, \gamma) = \frac{1}{1 + e^{-\gamma^T x}} \]

\[ b(x, \gamma) = \gamma_1 1\{x_3 \leq \gamma_2\} \]

**Algorithm 10.2 Forward Stagewise Additive Modeling.**

1. Initialize \( f_0(x) = 0 \).
2. For \( m = 1 \) to \( M \):
   
   (a) Compute

   \[ (\beta_m, \gamma_m) = \arg \min_{\beta, \gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)). \]

   (b) Set \( f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m) \).

Idea: greedily add one function at a time
Forward Stagewise Additive models

\[ b(x, \gamma) \text{ is a function with parameters } \gamma \]

Examples:

\[ b(x, \gamma) = \frac{1}{1 + e^{-\gamma^T x}} \]

\[ b(x, \gamma) = \gamma_1 1\{x_3 \leq \gamma_2\} \]

**Algorithm 10.2 Forward Stagewise Additive Modeling.**

1. Initialize \( f_0(x) = 0 \).
2. For \( m = 1 \) to \( M \):
   
   (a) Compute
   
   \[ (\beta_m, \gamma_m) = \arg \min_{\beta, \gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)). \]
   
   (b) Set \( f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m). \)

Idea: greedily add one function at a time

**AdaBoost:** \( b(x, \gamma) \): classifiers to \( \{-1, 1\} \)

\[ L(y, f(x)) = \exp(-y f(x)) \]
Forward Stagewise Additive models

$b(x, \gamma)$ is a function with parameters $\gamma$

Examples:

$$b(x, \gamma) = \frac{1}{1 + e^{-\gamma^T x}}$$

$$b(x, \gamma) = \gamma_1 1\{x_3 \leq \gamma_2\}$$

**Algorithm 10.2 Forward Stagewise Additive Modeling.**

1. Initialize $f_0(x) = 0$.
2. For $m = 1$ to $M$:
   
   (a) Compute
   $$\left(\beta_m, \gamma_m\right) = \arg\min_{\beta, \gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)).$$
   
   (b) Set $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$.

Idea: greedily add one function at a time

**Boosted Regression Trees:**

$$L(y, f(x)) = (y - f(x))^2$$

$b(x, \gamma)$: regression trees
Forward Stagewise Additive models

\[ b(x, \gamma) \] is a function with parameters \( \gamma \)

**Examples:**

\[ b(x, \gamma) = \frac{1}{1 + e^{-\gamma^T x}} \]

\[ b(x, \gamma) = \gamma_1 1\{x_3 \leq \gamma_2\} \]

**Algorithm 10.2 Forward Stagewise Additive Modeling.**

1. Initialize \( f_0(x) = 0 \).
2. For \( m = 1 \) to \( M \):
   
   (a) Compute
   \[
   (\beta_m, \gamma_m) = \arg \min_{\beta, \gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)).
   \]
   
   (b) Set \( f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m) \).

**Idea:** greedily add one function at a time

**Boosted Regression Trees:**

\[ L(y, f(x)) = (y - f(x))^2 \]

\[ L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)) = (y_i - f_{m-1}(x_i) - \beta b(x_i; \gamma))^2 \]

\[ = (r_{im} - \beta b(x_i; \gamma))^2, \quad r_{im} = y_i - f_{m-1}(x_i) \]

**Efficient:** No harder than learning regression trees!
Forward Stagewise Additive models

\[ b(x, \gamma) \text{ is a function with parameters } \gamma \]

Examples:

\[ b(x, \gamma) = \frac{1}{1 + e^{-\gamma^T x}} \]

\[ b(x, \gamma) = \gamma_1 1\{x_3 \leq \gamma_2\} \]

**Algorithm 10.2** Forward Stagewise Additive Modeling.

1. Initialize \( f_0(x) = 0 \).
2. For \( m = 1 \) to \( M \):
   
   (a) Compute
   
   \[
   (\beta_m, \gamma_m) = \arg \min_{\beta, \gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)).
   \]
   
   (b) Set \( f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m) \).

Idea: greedily add one function at a time

**Boosted Logistic Trees:**

\[
L(y, f(x)) = y \log(f(x)) + (1 - y) \log(1 - f(x))
\]

\( b(x, \gamma) \): regression trees

Computationally hard to update
Gradient Boosting

Least squares, exponential loss easy. But what about cross entropy? Huber?

Algorithm 10.3 Gradient Tree Boosting Algorithm.

1. Initialize $f_0(x) = \arg \min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.
2. For $m = 1$ to $M$:
   (a) For $i = 1, 2, \ldots, N$ compute
   $$r_{im} = - \left[ \frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f=f_{m-1}}.$$
   (b) Fit a regression tree to the targets $r_{im}$ giving terminal regions $R_{jm}$, $j = 1, 2, \ldots, J_m$.
   (c) For $j = 1, 2, \ldots, J_m$ compute
   $$\gamma_{jm} = \arg \min_\gamma \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma).$$
   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. Output $\hat{f}(x) = f_M(x)$.

XGBoost

LS fit regression tree to n-dimensional gradient, take a step in that direction
Least squares, 0/1 loss easy. But what about cross entropy? Huber?

AdaBoost uses 0/1 loss, all other trees are minimizing binomial deviance
Additive models

• Boosting is popular at parties: Invented by theorists, heavily adopted by practitioners.
Additive models

• Boosting is popular at parties: Invented by theorists, heavily adopted by practitioners.

• Computationally efficient with “weak” learners. But can also use trees! Boosting can scale.

• Kind of like sparsity?
Additive models

• Boosting is popular at parties: Invented by theorists, heavily adopted by practitioners.

• Computationally efficient with “weak” learners. But can also use trees! Boosting can scale.

• Kind of like sparsity?

• Gradient boosting generalization with good software packages (e.g., XGBoost). Effective on Kaggle

• Robust to overfitting and can be dealt with with “shrinkage” and “sampling”
Bagging versus Boosting

- Bagging *averages* many low-bias, lightly dependent classifiers to reduce the variance.
- Boosting *learns* linear combination of high-bias, highly dependent classifiers to reduce error.
- Empirically, boosting appears to outperform bagging.