Gradient Boosting for Ordinal Classification

Kemeng (Coleman) Zhang

Department of Applied Mathematics & Statistics
Johns Hopkins University

April 28, 2020

Preamble

- We study the problem of ordinal classification - response is from a set of finite, **discrete**, and **ordered** class labels
- The problem shares some similarities to **multi-classification** and **regression**
- In contrast to the former: the **order** between class labels **cannot be neglected**
- In contrast to the latter: the **scale** of the label is **not cardinal**

Preamble

- We study the problem of ordinal classification - response is from a set of finite, **discrete**, and **ordered** class labels
- The problem shares some similarities to **multi-classification** and **regression**
- In contrast to the former: the **order** between class labels **cannot be neglected**
- In contrast to the latter: the **scale** of the label is **not cardinal**

Preamble

- We study the problem of ordinal classification - response is from a set of finite, **discrete**, and **ordered** class labels
- The problem shares some similarities to **multi-classification** and **regression**
- In contrast to the former: the **order** between class labels **cannot be neglected**
- In contrast to the latter: the **scale** of the label is **not cardinal**

Preamble

- We study the problem of ordinal classification - response is from a set of finite, **discrete**, and **ordered** class labels
- The problem shares some similarities to **multi-classification** and **regression**
- In contrast to the former: the **order** between class labels **cannot be neglected**
- In contrast to the latter: the **scale** of the label is **not cardinal**

Preamble

- An algorithm based on **gradient tree boosting** is introduced
- Experiment results show **statistically significant** improvement over existing modeling approaches such as multi-classification and regression

Preamble

- An algorithm based on **gradient tree boosting** is introduced
- Experiment results show **statistically significant** improvement over existing modeling approaches such as multi-classification and regression

Preamble

Applications

Ordinal classification problems are **prevalent** in real-world applications

- **Collaborative Filtering & Recommendation** Predict user ratings on a likert scale and recommend new items
- **Priority filtering** classify emails to ordered groups; prioritize patients in need of urgent care

Preamble

Applications

Ordinal classification problems are **prevalent** in real-world applications

- **Collaborative Filtering & Recommendation** Predict user ratings on a likert scale and recommend new items
- **Priority filtering** classify emails to ordered groups; prioritize patients in need of urgent care

Plan for today...

Ordinal Classification

Existing Approaches

- The most popular approach to date is the Proportional odds model [McCullagh, 1980], often called the cumulative logits
- Shortcomings: likelihood objective is complex, and hard to optimize, numerically unstable
- Not clear how to incorporate non-linearity and ensure non-decreasing intercept terms

Ordinal Classification

Existing Approaches

- The most popular approach to date is the Proportional odds model [McCullagh, 1980], often called the cumulative logits
- Shortcomings: likelihood objective is complex, and hard to optimize, numerically unstable
- Not clear how to incorporate non-linearity and ensure non-decreasing intercept terms

Ordinal Classification

Existing Approaches

- The most popular approach to date is the Proportional odds model [McCullagh, 1980], often called the cumulative logits
- Shortcomings: likelihood objective is complex, and hard to optimize, numerically unstable
- Not clear how to incorporate non-linearity and ensure non-decreasing intercept terms

Ordinal Classification

Supervised Learning

- Consider random variables $(X, Y) \sim F_{X,Y}$, $X \in \mathcal{X}$ input features, $Y \in \mathcal{Y}$ response
- Given loss function $\ell : \mathcal{A} \times \mathcal{Y} \to [0, \infty)$
- Find hypothesis $\eta : \mathcal{X} \to \mathcal{A}$ that minimizes expected loss, or *risk* $R(\eta) = \mathbb{E}_{X,Y}\ell(\eta(X), Y)$
- We have access to a database of n observations drawn i.i.d from $F_{X,Y}$: $\mathcal{D}_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$
- Empirical risk function $\hat{R}_n(\eta) = \frac{1}{n}\sum_{i=1}^n \ell(\eta(\mathbf{x}_i), y_i)$
- Empirical risk minimization over hypothesis space $\mathcal{F}$

$$\hat{\eta}_n = \underset{\eta \in \mathcal{F}}{\operatorname{argmin}}\ \hat{R}_n(\eta)$$

Ordinal Classification
Supervised Learning

- Consider random variables $(X, Y) \sim F_{X,Y}$, $X \in \mathcal{X}$ input features, $Y \in \mathcal{Y}$ response
- Given loss function $\ell : \mathcal{A} \times \mathcal{Y} \to [0, \infty)$
- Find hypothesis $\eta : \mathcal{X} \to \mathcal{A}$ that minimizes expected loss, or *risk* $R(\eta) = \mathbb{E}_{X,Y}\ell(\eta(X), Y)$
- We have access to a database of n observations drawn i.i.d from $F_{X,Y}$: $\mathcal{D}_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$
- Empirical risk function $\hat{R}_n(\eta) = \frac{1}{n}\sum_{i=1}^n \ell(\eta(\mathbf{x}_i), y_i)$
- Empirical risk minimization over hypothesis space $\mathcal{F}$

$$\hat{\eta}_n = \operatorname*{argmin}_{\eta \in \mathcal{F}} \hat{R}_n(\eta)$$

Ordinal Classification

Supervised Learning

- Consider random variables $(X, Y) \sim F_{X,Y}$, $X \in \mathcal{X}$ input features, $Y \in \mathcal{Y}$ response
- Given loss function $\ell : \mathcal{A} \times \mathcal{Y} \to [0, \infty)$
- Find hypothesis $\eta : \mathcal{X} \to \mathcal{A}$ that minimizes expected loss, or *risk* $R(\eta) = \mathbb{E}_{X,Y} \ell(\eta(X), Y)$
- We have access to a database of n observations drawn i.i.d from $F_{X,Y}$: $\mathcal{D}_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$
- Empirical risk function $\hat{R}_n(\eta) = \frac{1}{n} \sum_{i=1}^n \ell(\eta(\mathbf{x}_i), y_i)$
- Empirical risk minimization over hypothesis space $\mathcal{F}$

$$\hat{\eta}_n = \operatorname*{argmin}_{\eta \in \mathcal{F}} \hat{R}_n(\eta)$$

Ordinal Classification

Supervised Learning

- Consider random variables $(X, Y) \sim F_{X,Y}$, $X \in \mathcal{X}$ input features, $Y \in \mathcal{Y}$ response
- Given loss function $\ell : \mathcal{A} \times \mathcal{Y} \to [0, \infty)$
- Find hypothesis $\eta : \mathcal{X} \to \mathcal{A}$ that minimizes expected loss, or *risk* $R(\eta) = \mathbb{E}_{X,Y}\ell(\eta(X), Y)$
- We have access to a database of n observations drawn i.i.d from $F_{X,Y}$: $\mathcal{D}_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$
- Empirical risk function $\hat{R}_n(\eta) = \frac{1}{n}\sum_{i=1}^n \ell(\eta(\mathbf{x}_i), y_i)$
- Empirical risk minimization over hypothesis space $\mathcal{F}$

$$\hat{\eta}_n = \operatorname*{argmin}_{\eta \in \mathcal{F}} \hat{R}_n(\eta)$$

Ordinal Classification

Supervised Learning

- Consider random variables $(X, Y) \sim F_{X,Y}$, $X \in \mathcal{X}$ input features, $Y \in \mathcal{Y}$ response
- Given loss function $\ell : \mathcal{A} \times \mathcal{Y} \to [0, \infty)$
- Find hypothesis $\eta : \mathcal{X} \to \mathcal{A}$ that minimizes expected loss, or *risk* $R(\eta) = \mathbb{E}_{X,Y} \ell(\eta(X), Y)$
- We have access to a database of n observations drawn i.i.d from $F_{X,Y}$: $\mathcal{D}_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$
- Empirical risk function $\hat{R}_n(\eta) = \frac{1}{n} \sum_{i=1}^n \ell(\eta(\mathbf{x}_i), y_i)$
- Empirical risk minimization over hypothesis space $\mathcal{F}$

$$\hat{\eta}_n = \operatorname*{argmin}_{\eta \in \mathcal{F}} \hat{R}_n(\eta)$$

Ordinal Classification

Supervised Learning

- Consider random variables $(X, Y) \sim F_{X,Y}$, $X \in \mathcal{X}$ input features, $Y \in \mathcal{Y}$ response
- Given loss function $\ell : \mathcal{A} \times \mathcal{Y} \to [0, \infty)$
- Find hypothesis $\eta : \mathcal{X} \to \mathcal{A}$ that minimizes expected loss, or *risk* $R(\eta) = \mathbb{E}_{X,Y}\ell(\eta(X), Y)$
- We have access to a database of n observations drawn i.i.d from $F_{X,Y}$: $\mathcal{D}_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$
- Empirical risk function $\hat{R}_n(\eta) = \frac{1}{n}\sum_{i=1}^n \ell(\eta(\mathbf{x}_i), y_i)$
- Empirical risk minimization over hypothesis space $\mathcal{F}$

$$\hat{\eta}_n = \underset{\eta \in \mathcal{F}}{\operatorname{argmin}}\ \hat{R}_n(\eta)$$

Ordinal Classification

Supervised Learning

- Consider random variables $(X, Y) \sim F_{X,Y}$, $X \in \mathcal{X}$ input features, $Y \in \mathcal{Y}$ response
- Given loss function $\ell : \mathcal{A} \times \mathcal{Y} \to [0, \infty)$
- Find hypothesis $\eta : \mathcal{X} \to \mathcal{A}$ that minimizes expected loss, or *risk* $R(\eta) = \mathbb{E}_{X,Y}\ell(\eta(X), Y)$
- We have access to a database of n observations drawn i.i.d from $F_{X,Y}$: $\mathcal{D}_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$
- Empirical risk function $\hat{R}_n(\eta) = \frac{1}{n}\sum_{i=1}^n \ell(\eta(\mathbf{x}_i), y_i)$
- Empirical risk minimization over hypothesis space $\mathcal{F}$

$$\hat{\eta}_n = \underset{\eta \in \mathcal{F}}{\operatorname{argmin}}\ \hat{R}_n(\eta)$$

Ordinal Classification
Outcome Space

- $Y \in \{r_1, ..., r_K\}$ discrete class labels
- $r_1 \prec ... \prec r_{K-1} \prec r_K$, where $\prec$ denotes the ordering relation between the labels
- Let us assume, WLOG, that $r_k = k - 1$

Ordinal Classification

Outcome Space

- $Y \in \{r_1, \ldots, r_K\}$ discrete class labels
- $r_1 \prec \ldots \prec r_{K-1} \prec r_K$, where $\prec$ denotes the ordering relation between the labels
- Let us assume, WLOG, that $r_k = k - 1$

Ordinal Classification

Outcome Space

- $Y \in \{r_1, ..., r_K\}$ discrete class labels
- $r_1 \prec ... \prec r_{K-1} \prec r_K$, where $\prec$ denotes the ordering relation between the labels
- Let us assume, WLOG, that $r_k = k - 1$

Ordinal Classification

Loss Function

Definition (Ordinal loss function)

A loss function ℓ is an ordinal loss function if it satisfy the following properties

- $(i, i) = \underset{j,k \in \{0,\dots,K-1\}}{\operatorname{argmin}} \ \ell(j, k) \ \forall i \in \{0, \dots, K-1\}$
- $\ell(i, j) \leq \ell(k, l)$ if $|i - j| \leq |k - l|$

Comment: for example, it is worse to mistakenly classify a 1-star rating with a 10-star than a 4-star with a 5-star

Ordinal Classification

Loss Function

Definition (Ordinal loss function)

A loss function ℓ is an ordinal loss function if it satisfy the following properties

- $(i, i) = \underset{j,k \in \{0,\ldots,K-1\}}{\operatorname{argmin}} \ell(j, k) \ \forall i \in \{0, \ldots, K-1\}$
- $\ell(i, j) \leq \ell(k, l)$ if $|i - j| \leq |k - l|$

Comment: for example, it is worse to mistakenly classify a 1-star rating with a 10-star than a 4-star with a 5-star

Ordinal Classification

Loss Function Construction

- Consider binary classification where $Y \in \pm 1$
- $\tilde{y}_i = \hat{\eta}_n(\mathbf{x}_i)$
- plug-in Bayes decision

$$\hat{y}_i = \hat{g}_n(\mathbf{x}_i) = \begin{cases} 1 & \text{if } \tilde{y}_i > 0 \\ -1 & \text{o.w.} \end{cases} \tag{1}$$

Ordinal Classification

Loss Function Construction

- Consider binary classification where $Y \in \pm 1$
- $\tilde{y}_i = \hat{\eta}_n(\mathbf{x}_i)$
- plug-in Bayes decision

$$\hat{y}_i = \hat{g}_n(\mathbf{x}_i) = \begin{cases} 1 & \text{if } \tilde{y}_i > 0 \\ -1 & \text{o.w.} \end{cases} \tag{1}$$

Ordinal Classification

Loss Function Construction

- Consider binary classification where $Y \in \pm 1$
- $\tilde{y}_i = \hat{\eta}_n(\mathbf{x}_i)$
- plug-in Bayes decision

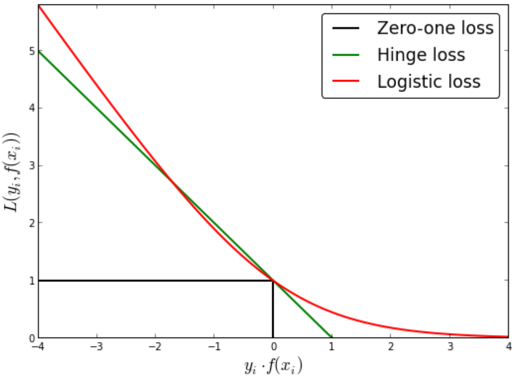
$$\hat{y}_i = \hat{g}_n(\mathbf{x}_i) = \begin{cases} 1 & \text{if } \tilde{y}_i > 0 \\ -1 & \text{o.w.} \end{cases} \tag{1}$$

Ordinal Classification

Loss Function Construction

Some examples of binary classification loss functions $l(\tilde{y}, y)$

- Zero-one loss: $\mathbb{1}_{\{\text{sgn}(\tilde{y}y) \neq 1\}}$
- Hinge loss: $\max(0, 1 - \tilde{y}y)$
- Logistic loss (Log loss): $\log(1 + \exp(-\tilde{y}y))$

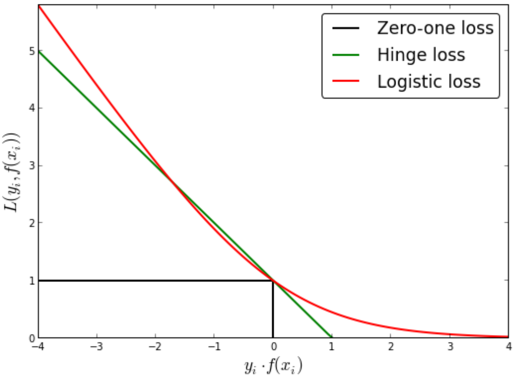


Ordinal Classification

Loss Function Construction

Some examples of binary classification loss functions $l(\tilde{y}, y)$

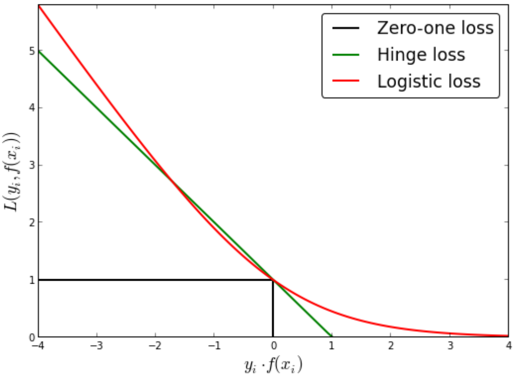
- Zero-one loss: $\mathbb{1}_{\{\text{sgn}(\tilde{y}y)\neq 1\}}$
- Hinge loss: $\max(0, 1 - \tilde{y}y)$
- Logistic loss (Log loss): $\log(1 + \exp(-\tilde{y}y))$



Ordinal Classification
Loss Function Construction

Some examples of binary classification loss functions $l(\tilde{y}, y)$

- Zero-one loss: $\mathbb{1}_{\{\mathrm{sgn}(\tilde{y}y)\neq 1\}}$
- Hinge loss: $\max(0, 1 - \tilde{y}y)$
- Logistic loss (Log loss): $\log(1 + \exp(-\tilde{y}y))$



Ordinal Classification

Loss Function Construction

Dilemma:

- In the binary case, we use a single threshold, namely 0, to divide the real line into two distinct decision segments
- In the non-binary case, it is not clear, from $\hat{\eta}_n$, how to derive a decision function $\hat{g}_n$ and what the threshold might be
- We would like to learn the threshold in a data-driven way

Ordinal Classification

Loss Function Construction

Dilemma:

- In the binary case, we use a single threshold, namely 0, to divide the real line into two distinct decision segments
- In the non-binary case, it is not clear, from $\hat{\eta}_n$, how to derive a decision function $\hat{g}_n$ and what the threshold might be
- We would like to learn the threshold in a data-driven way

Ordinal Classification

Loss Function Construction

Dilemma:

- In the binary case, we use a single threshold, namely 0, to divide the real line into two distinct decision segments
- In the non-binary case, it is not clear, from $\hat{\eta}_n$, how to derive a decision function $\hat{g}_n$ and what the threshold might be
- We would like to learn the threshold in a data-driven way

Ordinal Classification

Loss Function Construction

Extending binary classification - Threshold-based construction

- Suppose we are given $K-1$ decision thresholds $\theta_0 < \theta_1 < ... < \theta_{K-2}$, denoted as $\boldsymbol{\theta} \in \mathbb{R}^{K-1}$
- Each of the K segment on the real line corresponds to one of the K labels

$$\hat{y}_i = \hat{g}_n(\mathbf{x}_i) = \begin{cases} 0 & \text{if } \tilde{y}_i \in (-\infty, \theta_0) \\ k & \text{if } \tilde{y}_i \in [\theta_{k-1}, \theta_k) \\ K-1 & \text{if } \tilde{y}_i \in [\theta_{K-2}, \infty) \end{cases} \tag{2}$$

Ordinal Classification

Loss Function Construction

Extending binary classification - Threshold-based construction

- Suppose we are given $K-1$ decision thresholds $\theta_0 < \theta_1 < ... < \theta_{K-2}$, denoted as $\boldsymbol{\theta} \in \mathbb{R}^{K-1}$
- Each of the K segment on the real line corresponds to one of the K labels

$$\hat{y}_i = \hat{g}_n(\mathbf{x}_i) = \begin{cases} 0 & \text{if } \tilde{y}_i \in (-\infty, \theta_0) \\ k & \text{if } \tilde{y}_i \in [\theta_{k-1}, \theta_k) \\ K-1 & \text{if } \tilde{y}_i \in [\theta_{K-2}, \infty) \end{cases} \tag{2}$$

Ordinal Classification

Loss Function Construction

All-Threshold loss function [Rennie et al. 2005]

- Sum up all threshold violations

$$\ell_{ord}(\tilde{y}_i, y_i, \boldsymbol{\theta}) = \sum_{k=0}^{y_i-1} \ell(\tilde{y}_i - \theta_k, 1) + \sum_{k=y_i}^{K-2} \ell(\tilde{y}_i - \theta_k, -1) \quad (3)$$

Ordinal Classification

Loss Function Construction

All-Threshold loss function [Rennie et al. 2005]

- Sum up all threshold violations

$$\ell_{ord}(\tilde{y}_i, y_i, \boldsymbol{\theta}) = \sum_{k=0}^{y_i - 1} \ell(\tilde{y}_i - \theta_k, 1) + \sum_{k=y_i}^{K-2} \ell(\tilde{y}_i - \theta_k, -1) \quad (3)$$

Ordinal Classification

Loss Function Construction

All-Threshold loss function [Rennie et al. 2005]

$$\mathrm{s}(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x \leq 0 \end{cases} \tag{4}$$

$$\ell_{ord}(\tilde{y}_i, y_i, \boldsymbol{\theta}) = \sum_{k=0}^{K-2} \ell(\tilde{y}_i - \theta_k, \mathrm{s}(y_i - k)) \tag{5}$$

Ordinal Classification

Loss Function Construction

All-Threshold loss function [Rennie et al. 2005]

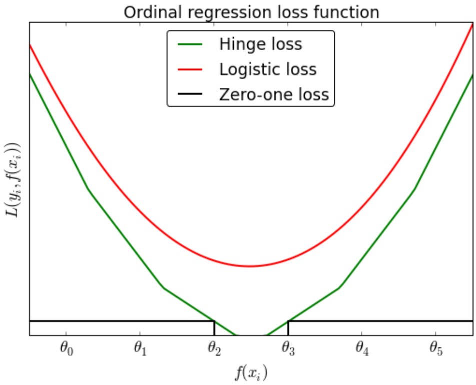


Figure: Ordinal Classification Loss Functions when $y_i = 3$.

Ordinal Classification

Loss Function Construction

Theorem (Ordered thresholds, Li et. al 2007)

If ℓ is convex, then

$$\exists\ \hat{\eta}_n, \boldsymbol{\theta}^* \in \underset{\eta \in \mathcal{F}, \boldsymbol{\theta}}{argmin} \frac{1}{n} \sum_{i=1}^{n} \ell_{ord}(\eta(\mathbf{x}_i), y_i, \boldsymbol{\theta})$$

such that $\boldsymbol{\theta}^*$ is ordered, i.e., $\theta_0 \leq \theta_1 \leq ... \leq \theta_{K-1}$.

Ordinal Classification

Loss Function Construction

Theorem (Fisher Consistency, Pedregosa et. al 2014)

If ℓ is convex, $\ell(\tilde{y}, \cdot)$ is differentiable at 0, and $\frac{\partial}{\partial \tilde{y}} \ell(\tilde{y}, \cdot)|_{\tilde{y}=0} < 0$, then All-Threshold loss is Fisher consistent.

Ordinal Classification

Loss Function Construction

Probabilistic interpretation of All-Threshold loss

- All-Threshold loss can be seen as the sum of negative log-likelihood with the logit link function

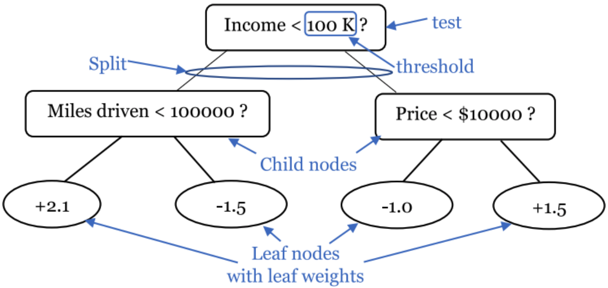
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$\psi(x) = \log(1+e^{-x}) = -\log(\sigma(x))$$

$$\mathbb{P}(y_i \leq k) = \sigma(\mathrm{s}(y_i - k)(\tilde{y}_i - \theta_k)) = \sigma(\theta_k - \tilde{y}_i)$$

Gradient Boosting

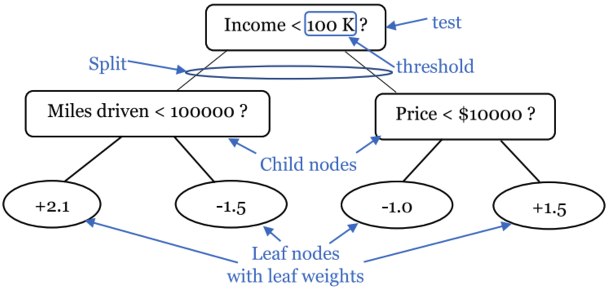
Building Block - Decision Trees



$$\tilde{y}_i = \sum_{j=1}^{J} w_j \mathbb{1}_{\{\mathbf{x}_i \in I_j\}}$$

Gradient Boosting

Building Block - Decision Trees



$$\tilde{y}_i = \sum_{j=1}^{J} w_j \mathbb{1}_{\{\mathbf{x}_i \in I_j\}}$$

Gradient Boosting
Building Block - Decision Trees

- Tree structure $q : \mathcal{X} \to \{1, 2, ..., J\}$
- Instance set $I_j = \{i; q(\mathbf{x}_i) = j\}$
- Output/leaf weight $f(\mathbf{x}) = w_{q(\mathbf{x})}$

Gradient Boosting

Building Block - Decision Trees

- Tree structure $q : \mathcal{X} \rightarrow \{1, 2, ..., J\}$
- Instance set $I_j = \{i; q(\mathbf{x}_i) = j\}$
- Output/leaf weight $f(\mathbf{x}) = w_{q(\mathbf{x})}$

Gradient Boosting

Building Block - Decision Trees

- Tree structure $q : \mathcal{X} \to \{1, 2, ..., J\}$
- Instance set $I_j = \{i; q(\mathbf{x}_i) = j\}$
- Output/leaf weight $f(\mathbf{x}) = w_{q(\mathbf{x})}$

Gradient Boosting

Regularized Learning Objective

- A tree ensemble model uses T additive functions to predict output

$$\tilde{y}_i = \sum_{t=1}^{T} f_t(\mathbf{x}_i), f_t \in \mathcal{F},$$

where $\mathcal{F} = \{f : f(\mathbf{x}) = w_{q(\mathbf{x})}\}$ is the space of regression trees (also known as CART).

- To learn the set of functions in the ensemble model, we minimize the following *regularized* objective:

$$\mathcal{L} = \sum_{i=1}^{n} \ell_{ord}(\tilde{y}_i, y_i, \boldsymbol{\theta}) + \sum_{t=1}^{T} \Omega(f_t) \tag{6}$$

where $\Omega(f) = \gamma J + \frac{1}{2}\lambda ||w||^2$

Gradient Boosting

Regularized Learning Objective

- A tree ensemble model uses T additive functions to predict output

$$\tilde{y}_i = \sum_{t=1}^{T} f_t(\mathbf{x}_i),\, f_t \in \mathcal{F},$$

where $\mathcal{F} = \{f : f(\mathbf{x}) = w_{q(\mathbf{x})}\}$ is the space of regression trees (also known as CART).

- To learn the set of functions in the ensemble model, we minimize the following *regularized* objective:

$$\mathcal{L} = \sum_{i=1}^{n} \ell_{ord}(\tilde{y}_i, y_i, \boldsymbol{\theta}) + \sum_{t=1}^{T} \Omega(f_t) \tag{6}$$

where $\Omega(f) = \gamma J + \frac{1}{2}\lambda||w||^2$

Gradient Boosting

Gradient Tree Boosting

- Let $\tilde{y}_i^{(t)}$ be the prediction of the i-th example at the t-th iteration, we need to add f_t, and $\boldsymbol{\delta}^{(t)}$ to minimize the following objective:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} \ell_{ord}(\tilde{y}_i^{(t-1)} + f_t(\mathbf{x}_i), y_i, \boldsymbol{\theta}^{(t-1)} + \boldsymbol{\delta}^{(t)}) + \Omega(f_t) \quad (7)$$

where $\tilde{y}_i^{(t-1)}$ is the combined output of the previous $t-1$ trees with the i-th data point as input.

Gradient Boosting
Gradient Tree Boosting

We initialize at $t = 0$:

$$\begin{aligned} \tilde{y}^{(0)}, \boldsymbol{\theta}^{(0)} &= \underset{\tilde{y}, \boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{n} \ell_{ord}(y_i, \tilde{y}, \boldsymbol{\theta}) \\ \tilde{y}_i^{(0)} &= \tilde{y}^{(0)} \quad \forall i = \{1, ..., n\} \end{aligned} \tag{8}$$

$$\begin{aligned} \mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} \Big[\ell_{ord}(y_i, \tilde{y}_i^{(t-1)}, \boldsymbol{\theta}^{(t-1)}) + g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i) \Big] + \\ (\sum_{i=1}^{n} \frac{\partial l_i}{\partial \boldsymbol{\theta}})^\intercal \boldsymbol{\delta}^{(t)} + \frac{1}{2} \boldsymbol{\delta}^{(t)\intercal} (\sum_{i=1}^{n} \frac{\partial^2 l_i}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\intercal}) \boldsymbol{\delta}^{(t)} + \\ (\sum_{i=1}^{n} f_t(\mathbf{x}_i) \frac{\partial^2 l_i}{\partial \boldsymbol{\theta} \partial \tilde{y}})^\intercal \boldsymbol{\delta}^{(t)} + \Omega(f_t) \end{aligned} \tag{9}$$

Gradient Boosting

Gradient Tree Boosting

$$\tilde{\mathcal{L}}^{(t)} = \sum_{j=1}^{J} \Big[\big(\sum_{i \in I_j} (g_i + \frac{\partial^2 l_i}{\partial \boldsymbol{\theta} \partial \tilde{y}}^{\intercal} \boldsymbol{\delta}^{(t)}) \big) w_j + \frac{1}{2} \big(\sum_{i \in I_j} h_i + \lambda \big) w_j^2 \Big] + \\ \big(\sum_{i=1}^{n} \frac{\partial l_i}{\partial \boldsymbol{\theta}} \big)^{\intercal} \boldsymbol{\delta}^{(t)} + \frac{1}{2} \boldsymbol{\delta}^{(t)\intercal} \big(\sum_{i=1}^{n} \frac{\partial^2 l_i}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^{\intercal}} \big) \boldsymbol{\delta}^{(t)} + \gamma J \tag{10}$$

where

$$l_i = \ell_{ord}(y_i, \tilde{y}, \boldsymbol{\theta})\big|_{\tilde{y}=\tilde{y}_i^{(t-1)}, \boldsymbol{\theta}=\boldsymbol{\theta}^{(t-1)}}$$

$$g_i = \frac{\partial}{\partial \tilde{y}} \ell_{ord}(y_i, \tilde{y}, \boldsymbol{\theta})\big|_{\tilde{y}=\tilde{y}_i^{(t-1)}, \boldsymbol{\theta}=\boldsymbol{\theta}^{(t-1)}}$$

$$h_i = \frac{\partial^2}{\partial \tilde{y}^2} \ell_{ord}(y_i, \tilde{y}, \boldsymbol{\theta})\big|_{\tilde{y}=\tilde{y}_i^{(t-1)}, \boldsymbol{\theta}=\boldsymbol{\theta}^{(t-1)}}$$

Gradient Boosting
Gradient Tree Boosting

- This looks scary, but this is just Newton-Raphson, and we can analyze it more easily in matrix form
- Trainable parameters (descent direction in a Newton-Raphson step):

$$\boldsymbol{\beta} = \begin{bmatrix} w_1 & \dots & w_J & \delta_0 & \dots & \delta_{K-2} \end{bmatrix}^\intercal$$

Full gradients:

$$\boldsymbol{g} = \begin{bmatrix} G_1 & \dots & G_J & \boldsymbol{u}^\intercal \end{bmatrix}^\intercal$$

Gradient Boosting

Gradient Tree Boosting

Hessian matrix:

$$\boldsymbol{H} = \begin{bmatrix} H_1 + \lambda & & 0 & — & \boldsymbol{L}_1^\intercal & — \\ & \ddots & & & \vdots & \\ 0 & & H_J + \lambda & — & \boldsymbol{L}_J^\intercal & — \\ | & & | & & & \\ \boldsymbol{L}_1 & \dots & \boldsymbol{L}_J & & \mathrm{V} & \\ | & & | & & & \end{bmatrix} \tag{11}$$

where $\tilde{\mathcal{L}}^{(t)}$ can be re-expressed in matrix form as:

$$\tilde{\mathcal{L}}^{(t)} = \boldsymbol{g}^\intercal \boldsymbol{\beta} + \frac{1}{2} \boldsymbol{\beta}^\intercal \boldsymbol{H} \boldsymbol{\beta} + \gamma J \tag{12}$$

Gradient Boosting

Gradient Tree Boosting

here

$$G_j = \sum_{i \in I_j} g_i \qquad \boldsymbol{u} = \sum_{i=1}^{n} \frac{\partial l_i}{\partial \boldsymbol{\theta}} \qquad H_j = \sum_{i \in I_j} h_i$$

$$\boldsymbol{L}_j = \sum_{i \in I_j} \frac{\partial^2 l_i}{\partial \boldsymbol{\theta} \partial \tilde{y}} \qquad \boldsymbol{V} = \sum_{i=1}^{n} \frac{\partial^2 l_i}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\intercal}$$

Gradient Boosting
Gradient Tree Boosting

$$\boldsymbol{\beta}^* = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\ \tilde{\mathcal{L}}^{(t)} = -\boldsymbol{H}^{-1}\boldsymbol{g} \tag{13}$$

The corresponding value for $\tilde{\mathcal{L}}^{(t)}$ is

$$\tilde{\mathcal{L}}^{(t)*} = -\frac{1}{2}\boldsymbol{g}^\intercal \boldsymbol{H}^{-1}\boldsymbol{g} + \gamma J \tag{14}$$

Gradient Boosting

Optimization - Speed Up Matrix Inverse

- But wait...
- Taking inverse $\boldsymbol{H}^{-1}$ is expensive!
- Time complexity $\mathcal{O}\big((J+K)^3\big)$
- Slows down algorithm when trees grow bigger!

Gradient Boosting

Optimization - Speed Up Matrix Inverse

- Observe the block structure of the Hessian matrix
- Block inverse formula readily available!

$$\boldsymbol{H}^{-1} = \begin{bmatrix} \boldsymbol{H}_\lambda^{-1} + \boldsymbol{H}_\lambda^{-1}\boldsymbol{L}^\intercal(\boldsymbol{V} - \boldsymbol{L}\boldsymbol{H}_\lambda^{-1}\boldsymbol{L}^\intercal)^{-1}\boldsymbol{L}\boldsymbol{H}_\lambda^{-1} & -\boldsymbol{H}_\lambda^{-1}\boldsymbol{L}^\intercal(\boldsymbol{V} - \boldsymbol{L}\boldsymbol{H}_\lambda^{-1}\boldsymbol{L}^\intercal)^{-1} \\ -(\boldsymbol{V} - \boldsymbol{L}\boldsymbol{H}_\lambda^{-1}\boldsymbol{L}^\intercal)^{-1}\boldsymbol{L}\boldsymbol{H}_\lambda^{-1} & (\boldsymbol{V} - \boldsymbol{L}\boldsymbol{H}_\lambda^{-1}\boldsymbol{L}^\intercal)^{-1} \end{bmatrix} \tag{15}$$

where

$$\boldsymbol{H}_\lambda = \begin{bmatrix} H_1 + \lambda & & \text{\Huge 0} \\ & \ddots & \\ \text{\Huge 0} & & H_J + \lambda \end{bmatrix}$$

Gradient Boosting

Optimization - Speed Up Matrix Inverse

- Plug back in eqn. (13) we get:

$$\boldsymbol{\delta}^{(t)*} = -\left(\boldsymbol{V} - \boldsymbol{L}\boldsymbol{H}_\lambda^{-1}\boldsymbol{L}^\intercal\right)^{-1}\left(\boldsymbol{u} - \boldsymbol{L}\boldsymbol{H}_\lambda^{-1}\boldsymbol{g}\right)$$

$$= -\left(\boldsymbol{V} - \sum_{j=1}^{J}\frac{\boldsymbol{L}_j\boldsymbol{L}_j^\intercal}{H_j+\lambda}\right)^{-1}\left(\boldsymbol{u} - \sum_{j=1}^{J}\frac{G_j}{H_j+\lambda}\boldsymbol{L}_j\right) \quad (16)$$

$$w_j^* = -\frac{G_j + \boldsymbol{L}_j^\intercal\boldsymbol{\delta}^{(t)*}}{H_j+\lambda}$$

Gradient Boosting

Optimization - Speed Up Matrix Inverse

- Even the quantity $(\boldsymbol{V} - \boldsymbol{L}\boldsymbol{H}_\lambda^{-1}\boldsymbol{L}^\intercal)^{-1}$ can be computed using rank 1 updates at each iteration (splitting one node in the decision tree).
- The key is the **Sherman-Morrison-Woodbury formula**

Gradient Boosting

Optimization - Speed Up Matrix Inverse

Before split (assuming the k^{th} node is our splitting candidate):

$$
\begin{aligned}
\left(\boldsymbol{V} - \boldsymbol{L}\boldsymbol{H}_{\boldsymbol{\lambda}}\boldsymbol{L}^\intercal\right)^{-1} &= (\boldsymbol{V} - \sum_{j=1}^{J} \frac{\boldsymbol{L}_j \boldsymbol{L}_j^\intercal}{H_j + \lambda})^{-1} \\
&= (\boldsymbol{V} - \sum_{j \neq k}^{J} \frac{\boldsymbol{L}_j \boldsymbol{L}_j^\intercal}{H_j + \lambda} - \frac{\boldsymbol{L}_k \boldsymbol{L}_k^\intercal}{H_k + \lambda})^{-1} \\
&= (\boldsymbol{A} - \frac{\boldsymbol{L}_k \boldsymbol{L}_k^\intercal}{H_k + \lambda})^{-1} \\
&= \boldsymbol{A}^{-1} + \frac{\boldsymbol{A}^{-1}\boldsymbol{L}_k \boldsymbol{L}_k^\intercal \boldsymbol{A}^{-1}}{H_k + \lambda - \boldsymbol{L}_k^\intercal \boldsymbol{A}^{-1} \boldsymbol{L}_k}
\end{aligned} \tag{17}
$$

Gradient Boosting

Optimization - Speed Up Matrix Inverse

After split:

$$\left(\boldsymbol{V} - \boldsymbol{L}\boldsymbol{H}_{\boldsymbol{\lambda}}\boldsymbol{L}^\intercal\right)^{-1} = (\boldsymbol{V} - \sum_{j \neq k}^{J} \frac{\boldsymbol{L}_j \boldsymbol{L}_j^\intercal}{H_j + \lambda} - \frac{\boldsymbol{L}_{k_1} \boldsymbol{L}_{k_1}^\intercal}{H_{k_1} + \lambda} - \frac{\boldsymbol{L}_{k_2} \boldsymbol{L}_{k_2}^\intercal}{H_{k_2} + \lambda})^{-1}$$

$$= (\boldsymbol{A} - \frac{\boldsymbol{L}_{k_1} \boldsymbol{L}_{k_1}^\intercal}{H_{k_1} + \lambda} - \frac{\boldsymbol{L}_{k_2} \boldsymbol{L}_{k_2}^\intercal}{H_{k_2} + \lambda})^{-1} \tag{18}$$

(and apply Sherman-Morrison-Woodbury twice) where

$$\boldsymbol{L}_k = \boldsymbol{L}_{k_1} + \boldsymbol{L}_{k_2}$$
$$H_k = H_{k_1} + H_{k_2}$$

Gradient Boosting

Optimization - Speed Up Matrix Inverse

- At the root node, the matrix $\boldsymbol{A}$ above is just the diagonal matrix $\boldsymbol{V}$, which is trivial to invert
- Hence we iteratively have access to the matrix inverse by invoking Sherman-Morrison each time.
- Now we are only paying the price of matrix multiplications!

Gradient Boosting

Optimization - Majorization Minimization

- Observe $\boldsymbol{H}$ is non-invertible when $\lambda = 0$: $\boldsymbol{H}\mathbf{1} = 0$
- Our objective is hard to optimize
- **Majorization Minimization** (MM) to the rescue!

Gradient Boosting

Optimization - Majorization Minimization

- Observe $\boldsymbol{H}$ is non-invertible when $\lambda = 0$: $\boldsymbol{H}\mathbf{1} = 0$
- Our objective is hard to optimize
- **Majorization Minimization** (MM) to the rescue!

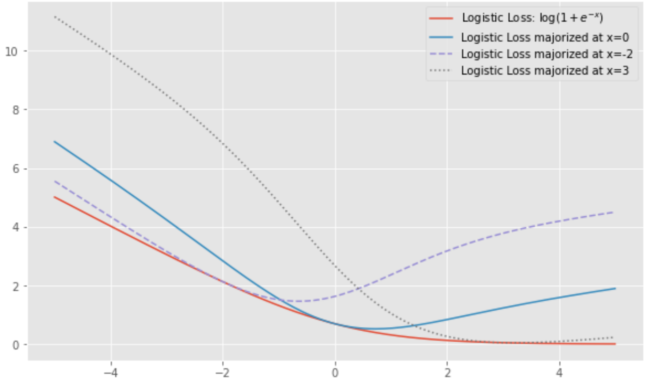
Gradient Boosting

Optimization - Majorization Minimization

- Observe $\boldsymbol{H}$ is non-invertible when $\lambda = 0$: $\boldsymbol{H}\mathbf{1} = 0$
- Our objective is hard to optimize
- **Majorization Minimization** (MM) to the rescue!

Gradient Boosting

Building Block - Decision Trees



Gradient Boosting

Optimization - Majorization Minimization

Definition (Majorization)

Suppose $\mathcal{L}$ and $\mathcal{M}$ are twice differentiable at $\mathbf{x}^{(t)}$. If $\mathcal{M}$ majorizes $\mathcal{L}$ at $\mathbf{x}^{(t)}$ then

- $\mathcal{M}(\mathbf{x}^{(t)}) = \mathcal{L}(\mathbf{x}^{(t)})$,
- $\frac{\partial \mathcal{M}}{\partial \mathbf{x}}\big|_{\mathbf{x}=\mathbf{x}^{(t)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}}\big|_{\mathbf{x}=\mathbf{x}^{(t)}}$
- $\frac{\partial^2 \mathcal{M}}{\partial \mathbf{x} \partial \mathbf{x}^\intercal}\big|_{\mathbf{x}=\mathbf{x}^{(t)}} \succ \frac{\partial^2 \mathcal{L}}{\partial \mathbf{x} \partial \mathbf{x}^\intercal}\big|_{\mathbf{x}=\mathbf{x}^{(t)}}$

Gradient Boosting

Optimization - Majorization Minimization

Theorem (Böhning & Lindsay 1988)

Let $\mathbf{x}^{(0)} \in \mathcal{X}$ *and suppose that* $(\mathbf{x}^{(t)})_{t \geq 1}$ *is defined by the Majorization Minimization algorithm*
$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \tilde{\boldsymbol{H}}(\mathbf{x}^{(t)})^{-1}\boldsymbol{g}(\mathbf{x}^{(t)})$, *where* $\tilde{\boldsymbol{H}}(\mathbf{x}) \succeq \boldsymbol{H}(\mathbf{x}) \; \forall \mathbf{x} \in \mathcal{X}$.
The sequence $(\mathbf{x}^{(t)})_{t \geq 1}$ *has the following properties:*

- ***Monotonicity****:* $\mathcal{L}(\mathbf{x}^{(t+1)}) \geq \mathcal{L}(\mathbf{x}^{(t)})$, *with strict inequality if* $\mathbf{x}^{(t+1)} \neq \mathbf{x}^{(t)}$.
- ***Guaranteed convergence****: The sequence* $\left(\boldsymbol{g}(\mathbf{x}^{(t)})\right)_{t \geq 1}$ *converges to 0 if* $\mathcal{L}$ *is bounded below.*
- ***Rate of convergence****: The algorithm converges with rate* $||\mathbb{I} - \tilde{\boldsymbol{H}}(\mathbf{x}^*)^{-1}\boldsymbol{H}(\mathbf{x}^*)||_2$.

Gradient Boosting

Optimization - Majorization Minimization

Quadratic Majorization:

$$\mathcal{M}(\mathbf{x}) = \mathcal{L}(\mathbf{x}^{(t)}) + \boldsymbol{g}(\mathbf{x}^{(t)})^\intercal(\mathbf{x} - \mathbf{x}^{(t)}) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^{(t)})^\intercal \tilde{\boldsymbol{H}}(\mathbf{x}^{(t)})(\mathbf{x} - \mathbf{x}^{(t)})$$

where $\tilde{\boldsymbol{H}}(\mathbf{x}) \succeq \boldsymbol{H}(\mathbf{x})\ \forall \mathbf{x} \in \mathcal{X}$

Gradient Boosting

Optimization - Majorization Minimization

Majorized Hessian $\tilde{\boldsymbol{H}}$ by majorizing the diagonal elements of $\boldsymbol{H}$

$$
\begin{aligned}
H_j + \lambda &= \sum_{i \in I_j} \sum_{k=0}^{K-2} \sigma(s_{ik} d_{ik}) \big(1 - \sigma(s_{ik} d_{ik})\big) + \lambda \\
\boldsymbol{V}_{kk} &= \sum_{i=1}^{n} \sigma(s_{ik} d_{ik}) \big(1 - \sigma(s_{ik} d_{ik})\big)
\end{aligned}
\tag{19}
$$

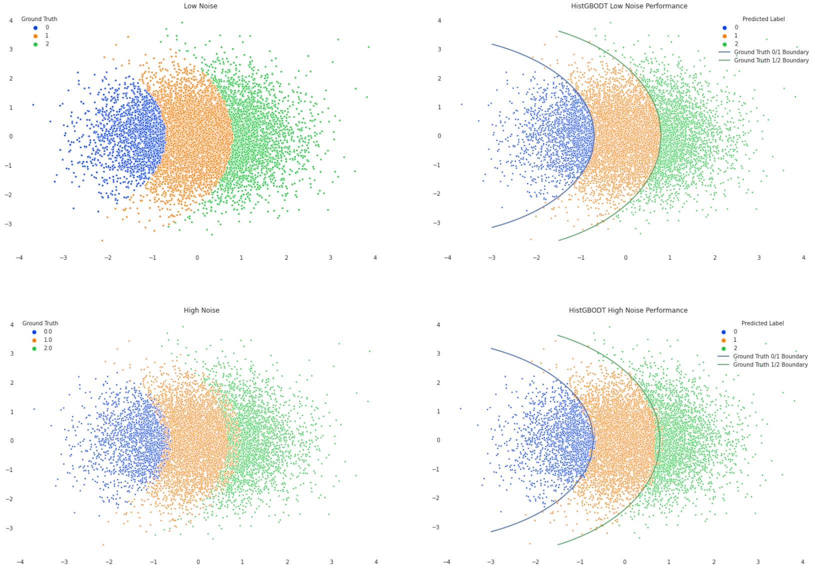
is replaced by:

$$
\begin{aligned}
\tilde{H}_j + \lambda &= \sum_{i \in I_j} \sum_{k=0}^{K-2} \frac{2\sigma(s_{ik} d_{ik}) - 1}{2 s_{ik} d_{ik}} + \lambda \\
\tilde{\boldsymbol{V}}_{kk} &= \sum_{i=1}^{n} \frac{2\sigma(s_{ik} d_{ik}) - 1}{2 s_{ik} d_{ik}}
\end{aligned}
\tag{20}
$$

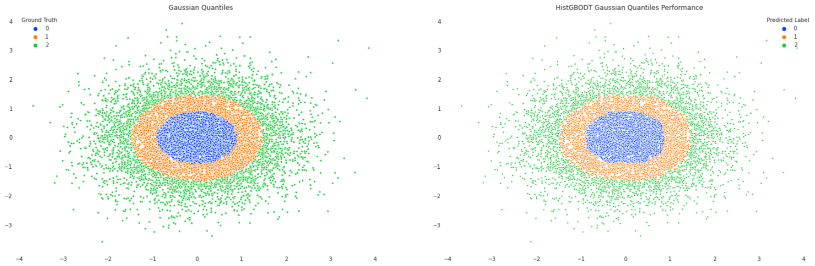
where $s_{ik} = \mathrm{s}(y_i - k)$, $d_{ik} = \tilde{y}_i - \theta_k$

Experiments

Artificial Data



Experiments
Artificial Data



`HistGBODT` performance on synthetic datasets.

The prediction accuracy on the three datasets are: 99.73%, 94.41%, and 98.01%, respectively.

Experiments

Benchmark Data

We performed experiments on 16 benchmark datasets that were used by Chu and Keerthi [1].

[1]Datasets available for download at:
http://www.gatsby.ucl.ac.uk/~chuwei/ordinalregression.html

Experiments
Benchmark Data

	Datasets	Attributes(Numeric,Nominal)	Training Instances	Instances for Test
	Diabetes	2(2,0)	30	13
	Pyrimidines	27(27,0)	50	24
	Triazines	60(60,0)	100	86
	Wisconsin Breast Cancer	32(32,0)	130	64
Set I	Machine CPU	6(6,0)	150	59
	Auto MPG	7(4,3)	200	192
	Boston Housing	13(12,1)	300	206
	Stocks Domain	9(9,0)	600	350
	Abalone	8(7,1)	3177	1000
	Bank Domains(1)	8(8,0)	5461	2731
	Bank Domains(2)	32(32,0)	5461	2731
	Computer Activity(1)	12(12,0)	5461	2731
Set II	Computer Activity(2)	21(21,0)	5461	2731
	Califomia Housing	8(8,0)	10427	5213
	Census Domains(1)	8(8,0)	11189	5595
	Census Domains(2)	16(16,0)	11189	5595

Table: Datasets and their characteristics. “Attributes” state the number of numerical and nominal attributes. “Training Instances” and “Instances for Test” specify the size of training/test partitions.

Experiments
Benchmark Data

	Mean zero-one error			Mean absolute Error		
Data	AT	MultiClass	MSE	AT	MultiClass	MSE
Diabetes	69.23%	69.23%	69.23%	1.154	1.154	1.154
Pyrimidines	**70.83%**	75.62%	77.92%	**1.240**	1.577	1.383
Triazines	67.85%	**67.79%**	71.05%	**1.112**	1.319	1.153
Wisconsin Breast Cancer	86.33%	**85.78%**	88.83%	**2.123**	2.805	2.438
Machine CPU	**31.69%**	41.10%	37.46%	**0.4483**	0.939	0.5161
Auto MPG	**45.83%**	50.86%	47.24%	**0.5464**	0.6331	0.5576
Boston Housing	**43.91%**	45.02%	44.22%	**0.5204**	0.5680	0.5235
Stocks Domain	22.06%	**21.29%**	21.84%	0.2261	**0.2164**	0.2257
Abalone	**42.34%**	48.40%	44.68%	**0.5212**	0.7552	0.5384
Bank Domains(1)	72.28%	**69.17%**	78.4%	**1.456**	1.788	1.530
Bank Domains(2)	**38.81%**	44.75%	39.47%	**0.4094**	0.5145	0.4171
Computer Activity(1)	**51.45%**	52.28%	57.66%	**0.7039**	0.8177	0.7929
Computer Activity(2)	**41.60%**	47.05%	45.18%	**0.4687**	0.5881	0.5134
Califomia Housing	**46.94%**	50.68%	49.18%	**0.5683**	0.6928	0.6016
Census Domains(1)	**60.58%**	62.59%	66.83%	**0.9296**	1.410	0.998
Census Domains(2)	62.25%	**60.25%**	66.93%	**0.9793**	1.130	1.022

Table: Testing set results: the bold font indicates the cases in which the average value is the lowest in the results of minimizing the three loss functions.

Experiments
Benchmark Data

	Mean zero-one error		Mean absolute error	
Null hypothesis	AT < MultiClass	AT < MSE	AT < MultiClass	AT < MSE
Test Statistic	20	1	1	1
P-value	0.01155	0.0004026	0.0004026	0.0004026

Table: Nonparametric Wilcoxon signed-rank test results

From the testing set results averaged over 20 trials, we could see a clear advantage of minimizing the All-Threshold loss over traditional methods such as multi-class cross entropy and mean-squared error. To determine statistical significance, we conduct Wilcoxon signed-rank tests. We observe that the performance gains are statistically significant for all comparisons.

Conclusion

and future directions...

- **Ordinal classification** is an important supervised learning task but has not been widely implemented in popular machine learning frameworks
- We propose minimizing the **All-Threshold** loss function combined with the powerful **gradient boosting** technique
- We address the issue of non-invertibility of the Hessian matrix using **Majorization-Minimization** (MM) to optimize objective
- Experiments on benchmark datasets indicates the generalization performance is competitive and **statistically significantly better** than existing approaches

Conclusion

and future directions...

- **Ordinal classification** is an important supervised learning task but has not been widely implemented in popular machine learning frameworks
- We propose minimizing the **All-Threshold** loss function combined with the powerful **gradient boosting** technique
- We address the issue of non-invertibility of the Hessian matrix using **Majorization-Minimization** (MM) to optimize objective
- Experiments on benchmark datasets indicates the generalization performance is competitive and **statistically significantly better** than existing approaches

Conclusion

and future directions...

- **Ordinal classification** is an important supervised learning task but has not been widely implemented in popular machine learning frameworks
- We propose minimizing the **All-Threshold** loss function combined with the powerful **gradient boosting** technique
- We address the issue of non-invertibility of the Hessian matrix using **Majorization-Minimization** (MM) to optimize objective
- Experiments on benchmark datasets indicates the generalization performance is competitive and **statistically significantly better** than existing approaches

Conclusion

and future directions...

- **Ordinal classification** is an important supervised learning task but has not been widely implemented in popular machine learning frameworks
- We propose minimizing the **All-Threshold** loss function combined with the powerful **gradient boosting** technique
- We address the issue of non-invertibility of the Hessian matrix using **Majorization-Minimization** (MM) to optimize objective
- Experiments on benchmark datasets indicates the generalization performance is competitive and **statistically significantly better** than existing approaches

Conclusion

and future directions...

- We implemented using `scikit-learn`, using `Cython` for native `C` speed performance and parallel computing. Would like to integrate with highly optimized gradient boosting package such as `XGBoost` and `CatBoost`
- Integration with ensemble of oblique decision trees

Conclusion

and future directions...

- We implemented using `scikit-learn`, using `Cython` for native `C` speed performance and parallel computing. Would like to integrate with highly optimized gradient boosting package such as `XGBoost` and `CatBoost`
- Integration with ensemble of oblique decision trees

Appendix

Gradient Statistics Full Expressions

$$G_j = \sum_{i \in I_j} g_i = \sum_{i \in I_j} \sum_{k=0}^{K-2} -s_{ik}\sigma(-s_{ik}d_{ik})$$

$$H_j = \sum_{i \in I_j} h_i = \sum_{i \in I_j} \sum_{k=0}^{K-2} \frac{2\sigma(s_{ik}d_{ik}) - 1}{2s_{ik}d_{ik}}$$

Appendix

Gradient Statistics Full Expressions

$$\boldsymbol{L}_j = \sum_{i \in I_j} \frac{\partial^2 l_i}{\partial \boldsymbol{\theta} \partial \tilde{y}} = \begin{bmatrix} \dots & \sum_{i \in I_j} -\sigma(s_{ik} d_{ik}) \big(1 - \sigma(s_{ik} d_{ik})\big) & \dots \end{bmatrix}^\intercal \in \mathbb{R}^{K-1}$$

$$\boldsymbol{u} = \sum_{i=1}^{n} \frac{\partial l_i}{\partial \boldsymbol{\theta}} = \sum_{i=1}^{n} \begin{bmatrix} \dots & s_{ik} \sigma(-s_{ik} d_{ik}) & \dots \end{bmatrix}^\intercal \in \mathbb{R}^{K-1}$$

$$\boldsymbol{V} = \sum_{i=1}^{n} \frac{\partial^2 l_i}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\intercal} = \operatorname{diag}\Big(\begin{bmatrix} \dots & \sum_{i=1}^{n} \frac{2\sigma(s_{ik} d_{ik}) - 1}{2 s_{ik} d_{ik}} & \dots \end{bmatrix} \Big) \in \mathbb{R}^{(K-1)\times(K-1)}$$