Ensembles II

Jiayu Zhou

[1]Department of Computer Science and Engineering
Michigan State University
East Lansing, MI USA

March 17, 2016

Slides from "A Gental Introduction to Gradient Boosting." by Cheng Li.

Table of contents

Introduction

Gradient Boosting

- A powerful machine learning algorithm
- Regression/Classification/Ranking.
- Won Track 1 of the Yahoo Learning to Rank Challenge

What is Gradient Boosting

- **Gradient Boosting = Gradient Descent + Boosting**
- AdaBoost

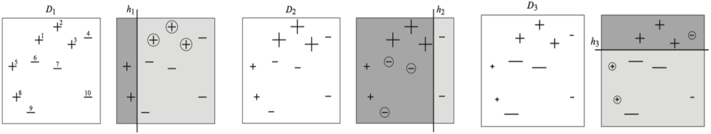


Figure : AdaBoost. Source: Figure 1.1 of [Schapire and Freund, 2012]

What is Gradient Boosting

- **Gradient Boosting = Gradient Descent + Boosting**
- AdaBoost

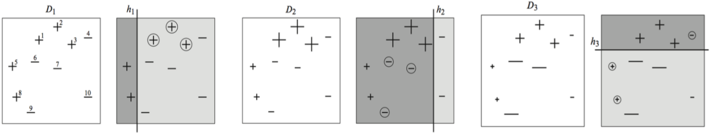


Figure : AdaBoost. Source: Figure 1.1 of [Schapire and Freund, 2012]

-
  - Fit an additive model (ensemble) $\sum_t \rho_t h_t(x)$ in a forward stage-wise manner.
  - In each stage, introduce a weak learner to compensate the shortcomings of existing weak learners.
  - In Adaboost, "shortcomings" are identified by high-weight data points.

What is Gradient Boosting

- **Gradient Boosting = Gradient Descent + Boosting**
- AdaBoost

$$H(x) = \sum_t \rho_t h_t(x)$$

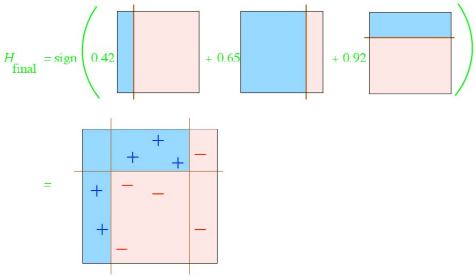


Figure : AdaBoost. Source: Figure 1.1 of [Schapire and Freund, 2012]

What is Gradient Boosting

- **Gradient Boosting = Gradient Descent + Boosting**
- Gradient Boosting
  - Fit an additive model (ensemble) $\sum_t \rho_t h_t(x)$ in a forward stage-wise manner.
  - In each stage, introduce a weak learner to compensate the shortcomings of existing weak learners.
    - In Gradient Boosting, shortcomings are identified by gradients.
    - In Adaboost, shortcomings are identified by high-weight data points.
  - Both high-weight data points and gradients tell us how to improve our model.

Regression

Gradient Boosting for Regression

Lets play a game...
You are given $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, and the task is to fit a model $F(x)$ to minimize square loss.
Suppose your friend wants to help you and gives you a model F. You check his model and find the model is good but not perfect. There are some mistakes: $F(x_1) = 0.8$, while $y_1 = 0.9$, and $F(x_2) = 1.4$ while $y2 = 1.3$ ... How can you improve this model?

Rule of the game:

- You are not allowed to remove anything from F or change any parameter in F.
- You can add an additional model (regression) h to F, so the new prediction will be $F(x) + h(x)$.

Gradient Boosting for Regression

Simple solution: You wish to improve the model such that

$$F(x_1) + h(x_1) = y_1$$
$$F(x_2) + h(x_2) = y_2$$
$$\cdots$$
$$F(x_n) + h(x_n) = y_n$$

Gradient Boosting for Regression

Simple solution: Or, equivalently, you wish

$$h(x_1) = y_1 - F(x_1)$$
$$h(x_2) = y_2 - F(x_2)$$
$$\dots$$
$$h(x_n) = y_n - F(x_n)$$

Can any regression model achieve goal perfectly?

Gradient Boosting for Regression

Simple solution: Or, equivalently, you wish

$$h(x_1) = y_1 - F(x_1)$$
$$h(x_2) = y_2 - F(x_2)$$
$$\dots$$
$$h(x_n) = y_n - F(x_n)$$

Can any regression model achieve goal perfectly?
Maybe not ...

Gradient Boosting for Regression

Simple solution: Or, equivalently, you wish

$$h(x_1) = y_1 - F(x_1)$$

$$h(x_2) = y_2 - F(x_2)$$

$$\dots$$

$$h(x_n) = y_n - F(x_n)$$

Can any regression model achieve goal perfectly?
Maybe not ...
But some regression model might be able to do this approximately.

Gradient Boosting for Regression

Simple solution: Or, equivalently, you wish

$$
\begin{aligned}
h(x_1) &= y_1 - F(x_1) \\
h(x_2) &= y_2 - F(x_2) \\
&\dots \\
h(x_n) &= y_n - F(x_n)
\end{aligned}
$$

Can any regression model achieve goal perfectly?
Maybe not ...
But some regression model might be able to do this approximately.
How?

Gradient Boosting for Regression

Simple solution: Or, equivalently, you wish

$$h(x_1) = y_1 - F(x_1)$$
$$h(x_2) = y_2 - F(x_2)$$
$$\ldots$$
$$h(x_n) = y_n - F(x_n)$$

Can any regression model achieve goal perfectly?
Maybe not ...
But some regression model might be able to do this approximately.
How?
Just fit a regression model h to data

$$(x_1, y_1 - F(x_1)), (x_2, y_2 - F(x_2)), ..., (x_n, y_n - F(x_n))$$

Gradient Boosting for Regression

Simple solution: Or, equivalently, you wish

$$h(x_1) = y_1 - F(x_1)$$
$$h(x_2) = y_2 - F(x_2)$$
$$\ldots$$
$$h(x_n) = y_n - F(x_n)$$

Can any regression model achieve goal perfectly?
Maybe not ...
But some regression model might be able to do this approximately.
How?
Just fit a regression model h to data

$$(x_1, y_1 - F(x_1)), (x_2, y_2 - F(x_2)), \ldots, (x_n, y_n - F(x_n))$$

Congratulations, you get a better model!

Gradient Boosting for Regression

Simple solution:

$y_i - F(x_i)$ are called residuals. These are the parts that existing model F cannot do well.

The role of h is to compensate the shortcoming of existing model F.

Gradient Boosting for Regression

Simple solution:

$y_i - F(x_i)$ are called residuals. These are the parts that existing model F cannot do well.

The role of h is to compensate the shortcoming of existing model F.

If the new model $F + h$ is still not satisfactory,

Gradient Boosting for Regression

Simple solution:

$y_i - F(x_i)$ are called residuals. These are the parts that existing model F cannot do well.

The role of h is to compensate the shortcoming of existing model F.

If the new model $F + h$ is still not satisfactory, we can add another regression model...

Gradient Boosting for Regression

Simple solution:

$y_i - F(x_i)$ are called residuals. These are the parts that existing model F cannot do well.

The role of h is to compensate the shortcoming of existing model F.

If the new model $F + h$ is still not satisfactory, we can add another regression model...

We are improving the predictions of training data, is the procedure also useful for test data?

Gradient Boosting for Regression

Simple solution:

$y_i - F(x_i)$ are called residuals. These are the parts that existing model F cannot do well.

The role of h is to compensate the shortcoming of existing model F.

If the new model $F + h$ is still not satisfactory, we can add another regression model...

We are improving the predictions of training data, is the procedure also useful for test data?

Yes! Because we are building a model, and the model can be applied to test data as well.

Gradient Boosting for Regression

Simple solution:

$y_i - F(x_i)$ are called residuals. These are the parts that existing model F cannot do well.

The role of h is to compensate the shortcoming of existing model F.

If the new model $F + h$ is still not satisfactory, we can add another regression model...

We are improving the predictions of training data, is the procedure also useful for test data?

Yes! Because we are building a model, and the model can be applied to test data as well.

How is this related to gradient descent?

Gradient Boosting for Regression

Gradient Descent: Minimize a function by moving in the opposite direction of the gradient.

$$\theta_i = \theta_i - \rho \frac{\partial J}{\partial \theta_i}, \quad \boldsymbol{\theta} = \boldsymbol{\theta} - \rho \nabla_{\boldsymbol{\theta}} J$$

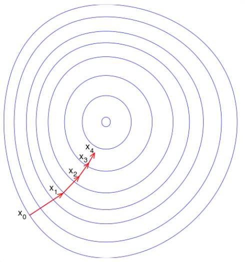


Figure: Gradient Descent.

Source: `http://en.wikipedia.org/wiki/Gradient_descent`

Gradient Boosting for Regression

How is this related to gradient descent?

Loss function $L(y, F(x)) = (yF(x))2/2$

We want to minimize $J = \sum_i L(y_i, F(x_i))$ by adjusting $F(x_1), F(x_2), \ldots, F(x_n)$.

Notice that $F(x_1), F(x_2), \ldots, F(x_n)$ are just some numbers. We can treat $F(x_i)$ as parameters and take derivatives.

Gradient Boosting for Regression

How is this related to gradient descent?

Loss function $L(y, F(x)) = (yF(x))2/2$

We want to minimize $J = \sum_i L(y_i, F(x_i))$ by adjusting $F(x_1), F(x_2), \dots, F(x_n)$.

Notice that $F(x_1), F(x_2), \dots, F(x_n)$ are just some numbers. We can treat $F(x_i)$ as parameters and take derivatives.

$$\frac{\partial \sum_i L(y_i, F(x_i))}{\partial F(x_i)} = \frac{\partial L(y_i, F(x_i))}{F(x_i)} = F(x_i) - y_i$$

So we can interpret residuals as negative gradients:

$$-\frac{\sum_i L(y_i, F(x_i))}{\partial F(x_i)} = y_i - F(x_i)$$

Gradient Boosting for Regression

How is this related to gradient descent?

$$F(x_i) = F(x_i) + h(x_i)$$
$$F(x_i) = F(x_i) + y_i - F(x_i)$$
$$F(x_i) = F(x_i) - 1\frac{\partial \sum_i L(y_i, F(x_i))}{\partial F(x_i)}$$
$$\theta_i = \theta_i - \rho\frac{\partial J}{\partial \theta_i}$$

Gradient Boosting for Regression

How is this related to gradient descent?

For regression with **square loss**,

$$\begin{aligned}
\text{residual} &= \text{negative gradient} \\
\text{fit h to residual} &= \text{fit h to negative gradient} \\
\text{update F based on residual} &= \text{update F based on negative gradient}
\end{aligned}$$

Gradient Boosting for Regression

How is this related to gradient descent?
For regression with **square loss**,

$$\begin{aligned}
\text{residual} &= \text{negative gradient} \\
\text{fit h to residual} &= \text{fit h to negative gradient} \\
\text{update F based on residual} &= \text{update F based on negative gradient}
\end{aligned}$$

So we are actually updating our model using **gradient descent**!

Gradient Boosting for Regression

How is this related to gradient descent?
For regression with **square loss**,

$$\begin{aligned} \text{residual} &= \text{negative gradient} \\ \text{fit h to residual} &= \text{fit h to negative gradient} \\ \text{update F based on residual} &= \text{update F based on negative gradient} \end{aligned}$$

So we are actually updating our model using **gradient descent**!
It turns out that the concept of **gradients** is more general and useful than the concept of **residuals**. So from now on, lets stick with gradients. The reason will be explained later.

Gradient Boosting for Regression

Let us summarize the algorithm we just derived using the concept of gradients. Negative gradient:

$$-g(x_i) = -\frac{\partial \sum_i L(y_i, F(x_i))}{\partial F(x_i)} = y_i - F(x_i)$$

1. start with an initial model, say, $F(x) = \frac{\sum_{i=1}^{n} y_i}{n}$
2. iterate until converge:
    - calculate negative gradients $-g(x_i)$
    - fit a regression model h to negative gradients $-g(x_i)$
    - $F = F + \rho h$, where $\rho = 1$

Gradient Boosting for Regression

Let us summarize the algorithm we just derived using the concept of gradients. Negative gradient:

$$-g(x_i) = -\frac{\partial \sum_i L(y_i, F(x_i))}{\partial F(x_i)} = y_i - F(x_i)$$

1. start with an initial model, say, $F(x) = \frac{\sum_{i=1}^{n} y_i}{n}$
2. iterate until converge:
   - calculate negative gradients $-g(x_i)$
   - fit a regression model h to negative gradients $-g(x_i)$
   - $F = F + \rho h$, where $\rho = 1$

The benefit of formulating this algorithm using gradients is that it allows us to consider other loss functions and derive the corresponding algorithms in the same way.

Gradient Boosting for Regression

Loss Functions for Regression Problem

Why do we need to consider other loss functions? Isnt square loss good enough?

Square loss is not robust to outliers

- Outliers are heavily punished because the error is squared.

y_i	0.5	1.2	2	5*
$F(x_i)$	0.6	1.4	1.5	1.7
$L = (y - F)^2/2$	0.005	0.02	0.125	5.445

- Consequence?

Gradient Boosting for Regression

Loss Functions for Regression Problem

Why do we need to consider other loss functions? Isnt square loss good enough?

Square loss is not robust to outliers

- Outliers are heavily punished because the error is squared.

y_i	0.5	1.2	2	5*
$F(x_i)$	0.6	1.4	1.5	1.7
$L = (y - F)^2/2$	0.005	0.02	0.125	5.445

- Consequence? Pay too much attention to outliers. Try hard to incorporate outliers into the model. Degrade the overall performance.

Gradient Boosting for Regression

Loss Functions for Regression Problem

- Absolute loss (more robust to outliers)

$$L(y, F) = |y - F|$$

Gradient Boosting for Regression

Loss Functions for Regression Problem

- Absolute loss (more robust to outliers)

$$L(y, F) = |y - F|$$

- Huber loss (more robust to outliers)

$$L(y, F) = \begin{cases} \frac{1}{2}(y - F)^2 & |y - F| \leq \delta \\ \delta(|y - F| - \delta/2) & |y - F| > \delta \end{cases}$$

Gradient Boosting for Regression

Loss Functions for Regression Problem

- Absolute loss (more robust to outliers)

$$L(y, F) = |y - F|$$

- Huber loss (more robust to outliers)

$$L(y, F) = \begin{cases} \frac{1}{2}(y - F)^2 & |y - F| \le \delta \\ \delta(|y - F| - \delta/2) & |y - F| > \delta \end{cases}$$

y_i	0.5	1.2	2	5*
$F(x_i)$	0.6	1.4	1.5	1.7
Square loss	0.005	0.02	0.125	5.445
Absolute loss	0.1	0.2	0.5	3.3
Huber loss ($\delta = 0.5$)	0.005	0.02	0.125	1.525

Gradient Boosting for Regression

Regression with Absolute Loss

Negative (sub)gradient

$$-g(x_i) = -\frac{\partial \sum_i L(y_i, F(x_i))}{\partial F(x_i)} = sign(y_i - F(x_i))$$

1. start with an initial model, say, $F(x) = \frac{\sum_{i=1}^{n} y_i}{n}$
2. iterate until converge:
   - calculate negative gradients $-g(x_i)$
   - fit a regression model h to negative gradients $-g(x_i)$
   - $F = F + \rho h$, where $\rho = 1$

Gradient Boosting for Regression

Regression with Huber Loss

Negative (sub)gradient

$$
\begin{aligned}
-g(x_i) &= -\frac{\partial \sum_i L(y_i, F(x_i))}{\partial F(x_i)} \\
&= \begin{cases} y_i - F(x_i) & |y_i - F(x_i)| \leq \delta \\ \delta \text{sign}(y_i - F(x_i)) & |y_i - F(x_i)| > \delta \end{cases}
\end{aligned}
$$

$$
-g(x_i) = -\frac{\partial \sum_i L(y_i, F(x_i))}{\partial F(x_i)} = sign(y_i - F(x_i))
$$

1. start with an initial model, say, $F(x) = \frac{\sum_{i=1}^{n} y_i}{n}$
2. iterate until converge:
   - calculate negative gradients $-g(x_i)$
   - fit a regression model h to negative gradients $-g(x_i)$
   - $F = F + \rho h$, where $\rho = 1$

Gradient Boosting for Regression

Regression with loss function L: general procedure

Given any (sub)differentiable loss function L

1. start with an initial model, say, $F(x) = \frac{\sum_{i=1}^{n} y_i}{n}$
2. iterate until converge:
   - calculate negative gradients $-g(x_i) = -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}$
   - fit a regression model h to negative gradients $-g(x_i)$
   - $F = F + \rho h$, where $\rho = 1$

Gradient Boosting for Regression

Regression with loss function L: general procedure
Given any (sub)differentiable loss function L

1. start with an initial model, say, $F(x) = \frac{\sum_{i=1}^{n} y_i}{n}$
2. iterate until converge:
   - calculate negative gradients $-g(x_i) = -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}$
   - fit a regression model h to negative gradients $-g(x_i)$
   - $F = F + \rho h$, where $\rho = 1$

In general,

$$\text{negative gradients} \neq \text{residuals}$$

We should follow negative gradients rather than residuals. Why?

Gradient Boosting for Regression

Negative Gradient vs Residual: An Example

Huber loss

$$L(y,F) = \begin{cases} \frac{1}{2}(y-F)^2 & |y-F| \leq \delta \\ \delta(|y-F| - \delta/2) & |y-F| > \delta \end{cases}$$

Update by Negative Gradient:

$$h(x_i) = -g(x_i) = \begin{cases} y_i - F(x_i) & |y_i - F(x_i)| \leq \delta \\ \delta \mathsf{sign}(y_i - F(x_i)) & |y_i - F(x_i)| > \delta \end{cases}$$

Update by Residual:

$$h(x_i) = y_i - F(x_i)$$

Difference: negative gradient pays less attention to outliers.

Gradient Boosting for Regression

Summary of the Section

- Fit an additive model $F = \sum_t \rho_t h_t$ in a forward stage-wise manner.
- In each stage, introduce a new regression tree h to compensate the shortcomings of existing model.
- The "shortcomings" are identified by negative gradients.
- For any loss function, we can derive a gradient boosting algorithm.
- Absolute loss and Huber loss are more robust to outliers than square loss.

Things not covered

How to choose a proper learning rate for each gradient boosting algorithm. See [Friedman, 2001]

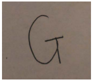
Classification

Gradient Boosting for Classification

Problem

Recognize the given hand written capital letter.

- Multi-class classification
- 26 classes. A,B,C,...,Z



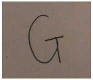
Data Set

- `http://archive.ics.uci.edu/ml/datasets/Letter+Recognition`
- 20000 data points, 16 features (how do we extract features in this case?)

Gradient Boosting for Classification

Feature Extraction

Statistical moments and edge counts



1	horizontal position of box	9	mean y variance
2	vertical position of box	10	mean x y correlation
3	width of box	11	mean of x * x * y
4	height of box	12	mean of x * y * y
5	total number on pixels	13	mean edge count left to right
6	mean x of on pixels in box	14	correlation of x-ege with y
7	mean y of on pixels in box	15	mean edge count bottom to top
8	mean x variance	16	correlation of y-ege with x

Feature Vector= (2,1,3,1,1,8,6,6,6,6,5,9,1,7,5,10)

Label = G

Gradient Boosting for Classification

Model

- 26 score functions (our models): $F_A, F_B, F_C, \ldots, F_Z$.
- $F_A(x)$ assigns a score for class A
- scores are used to calculate probabilities

$$P_A(x) = \frac{e^{F_A(x)}}{\sum_{c=A}^{Z} e^{F_c(x)}}$$

$$P_B(x) = \frac{e^{F_B(x)}}{\sum_{c=A}^{Z} e^{F_c(x)}}$$

$$\ldots$$

$$P_Z(x) = \frac{e^{F_Z(x)}}{\sum_{c=A}^{Z} e^{F_c(x)}}$$

- predicted label = class that has the highest probability

Loss Function for each data point

1. turn the label y_i into a (true) probability distribution $Y_c(x_i)$
   For example $y_5 = G$, then

$$Y_A(x_5) = 0, Y_B(x_5) = 0, \ldots, Y_G(x_5) = 1, \ldots, Y_Z(x_5) = 0$$

Loss Function for each data point

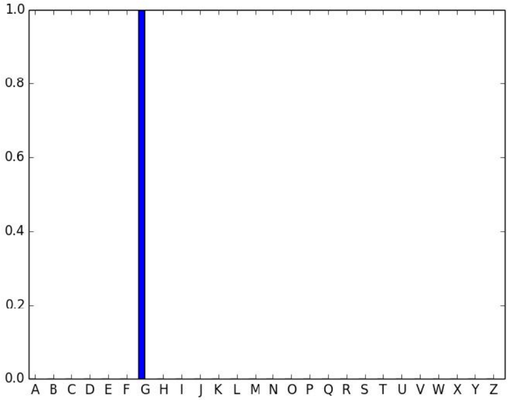


Figure: true probability distribution

Loss Function for each data point

1. turn the label y_i into a (true) probability distribution $Y_c(x_i)$
   For example $y_5 = G$, then

$$Y_A(x_5) = 0, Y_B(x_5) = 0, \dots, Y_G(x_5) = 1, \dots, Y_Z(x_5) = 0$$

2. calculate the predicted probability distribution $P_c(x_i)$ based on the current model $F_A, F_B, \dots, F_Z$.

$$P_A(x_5) = 0.03, P_B(x_5) = 0.05, \dots, P_G(x_5) = 0.3, \dots, PZ(x_5) = 0.05$$

Loss Function for each data point

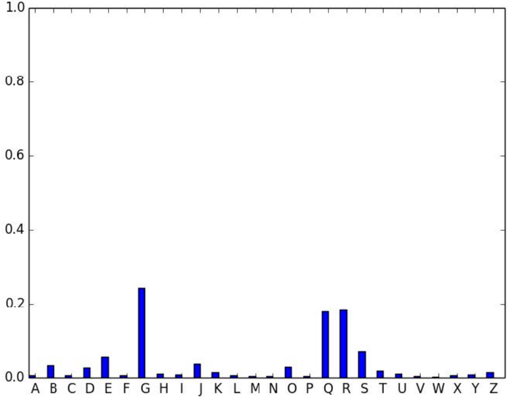


Figure: predicted probability distribution based on current model

Loss Function for each data point

1. turn the label y_i into a (true) probability distribution $Y_c(x_i)$
   For example $y_5 = G$, then

$$Y_A(x_5) = 0, Y_B(x_5) = 0, \ldots, Y_G(x_5) = 1, \ldots, Y_Z(x_5) = 0$$

2. calculate the predicted probability distribution $P_c(x_i)$ based on the current model $F_A, F_B, \ldots, F_Z$.

$$P_A(x_5) = 0.03, P_B(x_5) = 0.05, \ldots, P_G(x_5) = 0.3, \ldots, PZ(x_5) = 0.05$$

3. calculate the difference between the true probability distribution and the predicted probability distribution. Here we use KL-divergence

Gradient Boosting for Classification

Goal

- minimize the total loss (KL-divergence)
- for each data point, we wish the predicted probability distribution to match the true probability distribution as closely as possible

Minimizing KL Divergence

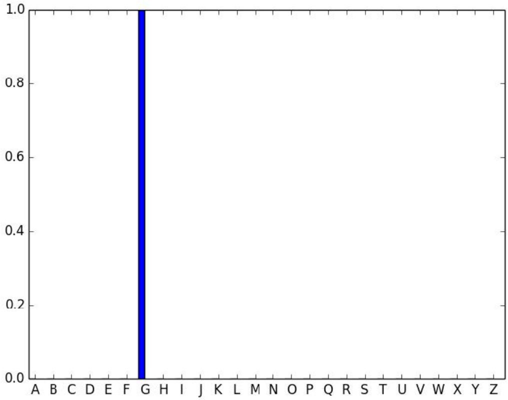


Figure: true probability distribution

Minimizing KL Divergence

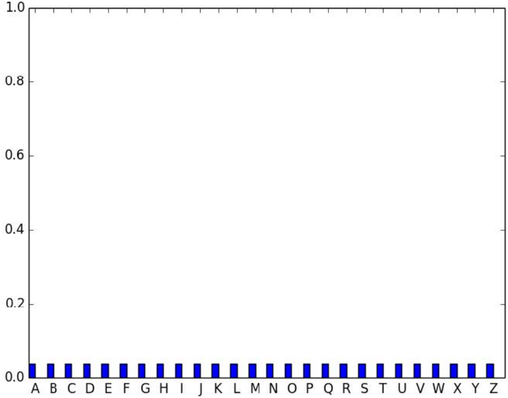


Figure: predicted probability distribution at round 0

Minimizing KL Divergence

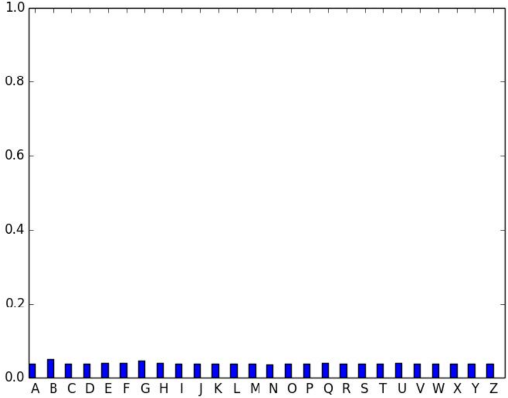


Figure: predicted probability distribution at round 1

Minimizing KL Divergence

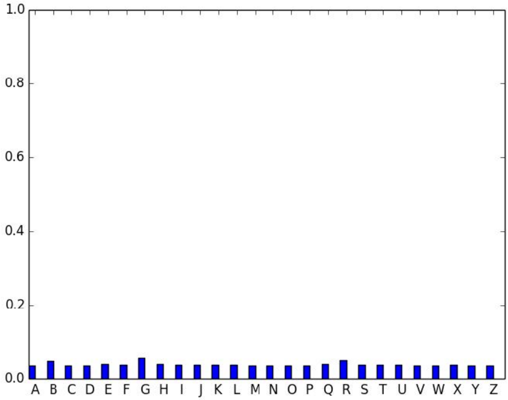


Figure: predicted probability distribution at round 2

Minimizing KL Divergence

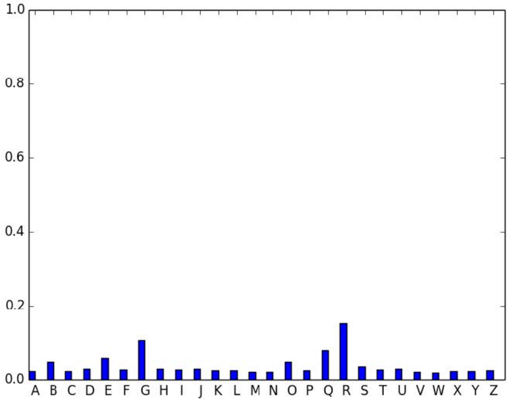


Figure: predicted probability distribution at round 10

Minimizing KL Divergence

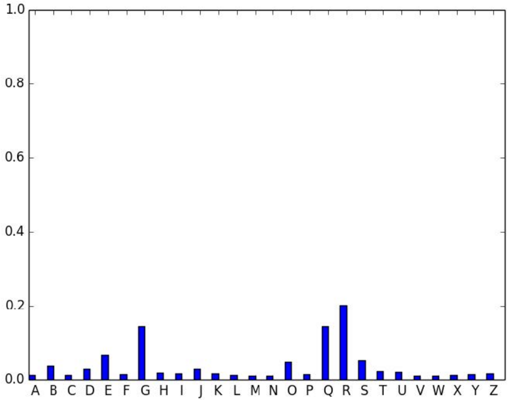


Figure: predicted probability distribution at round 20

Minimizing KL Divergence

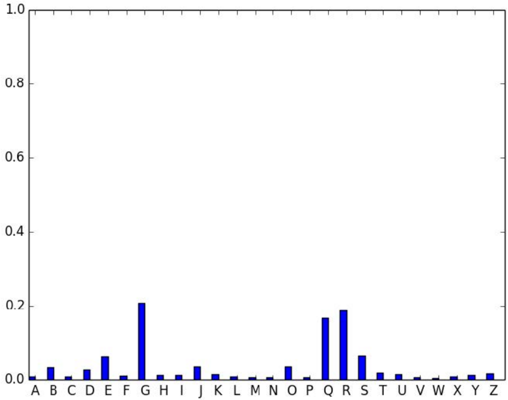


Figure: predicted probability distribution at round 30

Minimizing KL Divergence

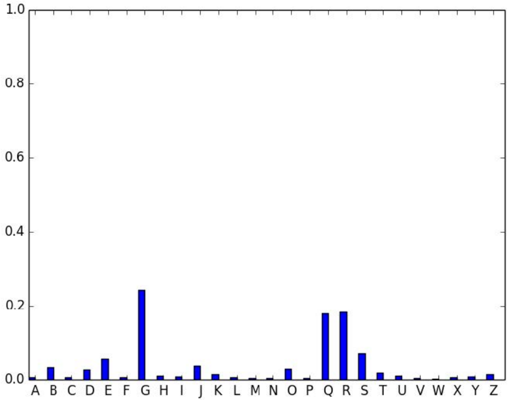


Figure: predicted probability distribution at round 40

Minimizing KL Divergence

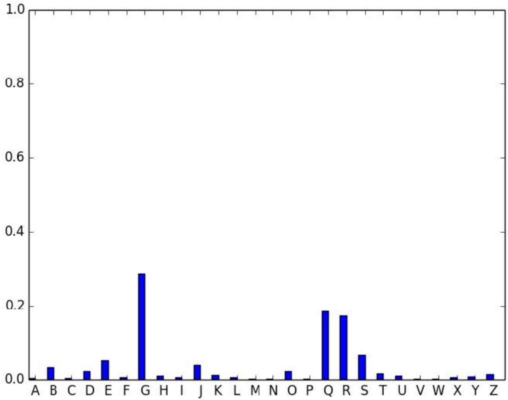


Figure: predicted probability distribution at round 50

Minimizing KL Divergence

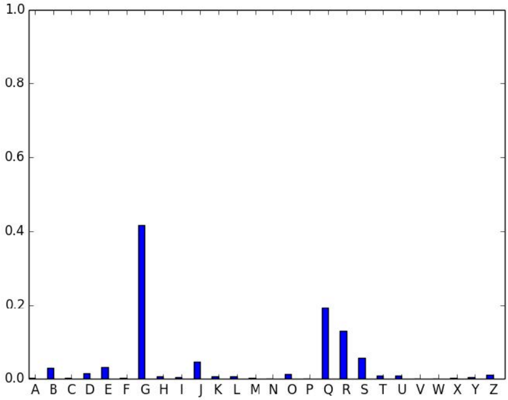


Figure: predicted probability distribution at round 100

Gradient Boosting for Classification

Goal

- minimize the total loss (KL-divergence)
- for each data point, we wish the predicted probability distribution to match the true probability distribution as closely as possible
- we achieve this goal by adjusting our models $F_A, F_B, \ldots, F_Z$.

Gradient Boosting for Regression: Review

Regression with loss function L: general procedure

Give any differentiable loss function L

start with an initial model F

iterate until converge:

- calculate negative gradients $-g(x_i) = -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}$
- fit a regression model h to negative gradients $-g(x_i)$
- $F = F + \rho h$

Gradient Boosting for Classification

Difference between classification and regression

- $F_A, F_B, \ldots, F_Z$ vs. F
- a matrix of parameters to optimize vs. a column of parameters to optimize

$F_A(x_1)$	$F_B(x_1)$	$\ldots$	$F_Z(x_1)$
$F_A(x_2)$	$F_B(x_2)$	$\ldots$	$F_Z(x_2)$
$\ldots$	$\ldots$	$\ldots$	$\ldots$
$F_A(x_n)$	$F_B(x_n)$	$\ldots$	$F_Z(x_n)$

- a matrix of gradients vs. a column of gradients

$\frac{\partial L}{\partial F_A(x_1)}$	$\frac{\partial L}{\partial F_B(x_1)}$	$\ldots$	$\frac{\partial L}{\partial F_Z(x_1)}$
$\frac{\partial L}{\partial F_A(x_2)}$	$\frac{\partial L}{\partial F_B(x_2)}$	$\ldots$	$\frac{\partial L}{\partial F_Z(x_2)}$
$\ldots$	$\ldots$	$\ldots$	$\ldots$
$\frac{\partial L}{\partial F_A(x_n)}$	$\frac{\partial L}{\partial F_B(x_n)}$	$\ldots$	$\frac{\partial L}{\partial F_Z(x_n)}$

Gradient Boosting for Classification

start with an initial models $F_A, F_B, F_C, \ldots, F_Z$
iterate until converge:

- calculate negative gradients for class A: $-g_A(x_i) = -\frac{\partial L(y_i, F(x_i))}{\partial F_A(x_i)}$
- calculate negative gradients for class B: $-g_B(x_i) = -\frac{\partial L(y_i, F(x_i))}{\partial F_B(x_i)}$
- ...
- calculate negative gradients for class Z: $-g_Z(x_i) = -\frac{\partial L(y_i, F(x_i))}{\partial F_Z(x_i)}$
- fit a regression model h_A to negative gradients $-g_A(x_i)$
- fit a regression model h_B to negative gradients $-g_B(x_i)$
- ...
- fit a regression model h_Z to negative gradients $-g_Z(x_i)$
- $F_A = F_A + \rho_A h_A$
- $F_B = F_B + \rho_B h_B$
- ...
- $F_Z = F_Z + \rho_Z h_Z$

Gradient Boosting for Classification

start with an initial models $F_A, F_B, F_C, \ldots, F_Z$
iterate until converge:

- calculate negative gradients for class A: $-g_A(x_i) = Y_A(x_i) - P_A(x_i)$
- calculate negative gradients for class B: $-g_B(x_i) = Y_B(x_i) - P_B(x_i)$
- $\ldots$
- calculate negative gradients for class Z: $-g_Z(x_i) = Y_Z(x_i) - P_Z(x_i)$
- fit a regression model h_A to negative gradients $-g_A(x_i)$
- fit a regression model h_B to negative gradients $-g_B(x_i)$
- $\ldots$
- fit a regression model h_Z to negative gradients $-g_Z(x_i)$
- $F_A = F_A + \rho_A h_A$
- $F_B = F_B + \rho_B h_B$
- $\ldots$
- $F_Z = F_Z + \rho_Z h_Z$

Gradient Boosting for Classification