**Ratchet:** The Underdog of Machine Learning Algorithms

Don Hush (work with Mike Cannon, Mike Fugate, and Clint Scovel at LANL)

2017 (origin 2002)
underdog:

- a competitor thought to have little chance of winning a fight or contest.
- *synonyms*: long shot, dark horse, weaker one, little guy, ...
- downtrodden, victim, loser, fall guy
  a person who has little status in society.
Summary:

- We present a very simple algorithm called Ratchet for optimizing empirical learning criteria $R$ that satisfy the PLD property.
- The Ratchet algorithm can be used in a large number of learning problems. We demonstrate two in detail and provide an extensive list of others.

Outline:

- Define the PLD property
- “Derive” the Ratchet algorithm
- How to Apply the Ratchet Algorithm (map original problem to PLD problem)
  - linear classifier for weighted 2-class classification
  - linear machine for weighted Multiclass Regression
- Examples

Note: The learning strategy we employ is empirical error minimization, which is (almost) always an NP-Hard problem. Ratchet is a randomized algorithm that finds an optimal solution asymptotically (with probability 1).
1. Let $Z = \{z_1, \ldots, z_n\}, z_i \in \mathbb{R}^m$ be a “data set” (strictly speaking it is a multiset).
2. $Z^+$ is PL subset of $Z$ if there exists an $\omega \in \mathbb{R}^m$ such that $\omega \cdot z_i > 0, \forall z_i \in Z^+$
3. The witness set $\Omega^+$ for a PL subset $Z^+$ is defined

$$\Omega^+ = \{\omega : \omega \cdot z_i > 0, \forall z_i \in Z^+\}$$

($\{z : \omega \cdot z = 0\}$ is a line through the origin)
The PLD Property

• Let $R$ be a real-valued function of the data set $Z$ and the parameter vector $\omega$. For example

$$ R(\omega) = |\{z_i : \omega \cdot z_i > 0\}| = \text{number of } \omega\text{-positive samples} $$

• Define the solution set

$$ \Omega^* = \arg\max_{\omega \in \mathbb{R}^m} R(\omega) $$

(we could choose the min instead of the max)

• The risk $R$ is PLD if there exists a PL subsample $Z^+ \subseteq Z$ whose witness set $\Omega^+$ satisfies $\Omega^+ \subseteq \Omega^*$.

• It is easy to prove that the example $R$ above is PLD:
  1. For any $\omega^* \in \Omega^*$ let $Z^*_+ = \{z_i : \omega^* \cdot z_i > 0\}$
  2. It is easy to prove that $\Omega^*_+ \subseteq \Omega^*$

Consequence of PLD Property: To maximize $R$ it is sufficient to witness a particular PL subset of $Z$. 
maximize the number of positively labeled points
Example #1

In this case $\Omega^+_\star = \Omega^*$
Example #2

maximize the number of positively labeled points
Example #2

Witness set $\Omega_1^+$. 
Example #2

Witness set $\Omega_2^+$. 
Example #2

Solution set $\Omega^* = \Omega_1^+ \cup \Omega_2^+$. 
Example #3

maximize the sum of point scores
Example #3

$Z^+$ does not always contain the largest number of points.
Example #4

maximize the sum of scores for complete groups (scores are 0 for other groups)
Example #4

$Z^+$ is not unique, individual points do not always contribute to $R$. 
maximize the sum of scores for complete or partial groups (scores are 0 for other groups)
$Z^+$ is not unique, individual points do not always contribute to $R$. 
Example: every sample $z$ has a score vector $(a_1, a_{-1})$ where

- $a_1 =$ score assigned if $z$ is on the positive side
- $a_{-1} =$ score assigned if $z$ is on the non-positive (negative) side

Goal: maximize the sum of sample scores
Example: every sample $z$ has a score vector $(a_1, a_{-1})$ where
- $a_1 = \text{score assigned if } z \text{ is on the positive side}$
- $a_{-1} = \text{score assigned if } z \text{ is on the non-positive (negative) side}$

Equivalent: replace $(a_1, a_{-1})$ by $\Delta = a_1 - a_{-1}$ and maximize sum of $\Delta$-scores as before
The Perceptron (PCP) Algorithm

INPUT: A “data set” \( Z = \{ z_1, z_2, \ldots, z_n \} \).

\[ \omega \leftarrow 0 \]

loop
  \( i \leftarrow \) next index from \( \{1, 2, \ldots, n\} \)
  if \( (\omega \cdot z_i \leq 0) \) then
    \( \omega \leftarrow \omega + z_i \)
  end if
end loop

Properties:

• If the entire set \( Z \) is PL then PCP will produce a member of the witness set \( \Omega^+ \) in a finite number of steps.

• If \( Z \) is not PL then PCP cycles endlessly ... (but \( \omega \) remains bounded)
The Randomized Perceptron (RP) Algorithm

**INPUT:** A “data set” $Z = \{z_1, z_2, ..., z_n\}$.

$\omega \leftarrow 0$

loop

  $i \leftarrow$ index chosen randomly from $\{1, 2, ..., n\}$ (uniform distribution)

  if $(\omega \cdot z_i \leq 0)$ then
  
    $\omega \leftarrow \omega + z_i$

  end if

end loop

**Properties:**

- If the entire set $Z$ is PL then RP will produce a member of the witness set $\Omega^+$ asymptotically (with probability 1).
- RP will produce a member of the witness set for every PL subset of $Z$ asymptotically (with probability 1).
The Ratchet Algorithm

INPUT: A “data set” $Z = \{z_1, z_2, ..., z_n\}$.

$\omega^* \leftarrow 0$, $R^* \leftarrow R(\omega^*)$
$\omega \leftarrow 0$

\{Perform the RP algorithm and track the best solution.\}

loop
    $i \leftarrow$ index chosen randomly from $\{1, 2, ..., n\}$
    if $(\omega \cdot z_i \leq 0)$ then
        $\omega \leftarrow \omega + z_i$
        if $(R(\omega) > R^*)$ then
            $R^* \leftarrow R(\omega)$
            $\omega^* \leftarrow \omega$
        end if
    end if
end loop

Theorem (Hush, 2002): If $R$ is PLD then $R(\omega^*) \xrightarrow{\text{wp1}} \max_\omega R(\omega)$. 
The Pocket Algorithm

INPUT: A “data set” $Z = \{z_1, z_2, \ldots, z_n\}$.

$\text{RunLength} \leftarrow 0$, $\text{MaxRunLength} \leftarrow 0$, $\omega \leftarrow 0$

{Perform the RP algorithm and keep $\omega$ that produces largest sequence of successes.}

\begin{algorithm}
\begin{algorithmic}
\State $i \leftarrow$ index chosen randomly from \{1, 2, \ldots, n\}
\If {$(\omega \cdot z_i > 0)$}
\State $\text{RunLength} \leftarrow \text{RunLength} + 1$
\Else\If {($\text{RunLength} > \text{MaxRunLength}$)}
\State $\text{MaxRunLength} \leftarrow \text{RunLength}$
\State $\omega^* \leftarrow \omega$
\EndIf
\State $\omega \leftarrow \omega + z_i$
\State $\text{RunLength} \leftarrow 0$
\EndIf
\EndIf
\end{algorithmic}
\end{algorithm}

Theorem (Gallant 1990, Muselli 1995): If $R = \#$ of positive samples then
\[ R(\omega^*) \overset{w.p.1}{\rightarrow} \max_{\omega} R(\omega). \]
Pocket vs Ratchet

- **Pocket:**
  - faster iterations than Ratchet
  - only valid for very specific criterion $R$

- **Ratchet:**
  - slower iterations than Pocket because it must evaluate $R$ each time $\omega$ is updated *
  - valid for any criterion $R$ that is PLD ... much more general!

* tricks for accelerating this evaluation
How Do We Use Ratchet?

What Do We Have to Do:

1. find a data map \( D \mapsto Z \) and
2. a criterion PLD \( R \) on \( Z \) that is calibrated to \( e \)
Example: Linear Classifier

**Problem Instance:** labeled data for a 2-class problem (with sample weights $a_i$)

\[ D = \{(x_1, y_1, a_1), \ldots, (x_n, y_n, a_n)\} \]
\[ x_i \in \mathbb{R}^d, \quad y_i \in \{-1, +1\}, \quad a_i \in \mathbb{R}^+ \]

**Linear Classifier** (parameterized by $\omega \in \mathbb{R}^{d+1}$):

\[ f_\omega(x) = \begin{cases} 
1, & \omega \cdot (1, x) > 0 \\
-1, & \omega \cdot (1, x) \leq 0 
\end{cases} \]

**Weighted Empirical Error:** (the weights satisfy $a_i > 0$)

\[ e(\omega) = \sum_{i:y_i=-1} a_i I(\omega \cdot (1, x_i) > 0) + \sum_{i:y_i=+1} a_i I(\omega \cdot (1, x_i) \leq 0) \]

**Goal:** find $\omega^*$ that minimizes $e$
dotted line shows possible solution when all the weights $a_i$ are equal
To simplify without obscuring the main point we consider only $\omega$ that satisfy

$$\omega \cdot (1, x_i) \neq 0, \ \forall x_i$$

*(all results hold without this restriction).* With this the error can be re-written

$$e(\omega) = \sum_{i:y_i=-1} a_i l(\omega \cdot (1, x_i) > 0) + \sum_{i:y_i=+1} a_i l(\omega \cdot (1, x_i) < 0)$$

$$= \sum_{i:y_i=-1} a_i l(\omega \cdot y_i(1, x_i) < 0) + \sum_{i:y_i=+1} a_i l(\omega \cdot y_i(1, x_i) < 0)$$

$$= \sum_{i=1}^n a_i l(\omega \cdot y_i(1, x_i) < 0)$$

$$= \sum_{i=1}^n a_i l(\omega \cdot z_i < 0)$$

where

$$z_i = y_i(1, x_i)$$

defines a data map.
Convert the inequality from $<$ to $>$:

\[ e(\omega) = \sum_{i=1}^{n} a_i I(\omega \cdot z_i < 0) \]

\[ = \sum_{i=1}^{n} a_i (1 - I(\omega \cdot z_i > 0)) \]

In this case it is trivial to prove that $e$ is PLD.

**Note:** Ratchet can minimize weighted classification error, but Pocket cannot.
Other Learning Problems

Learning problems that can be mapped to a surrogate weighted 2-class problem.

- Supervised Classification
- Anomaly Detection
- 1-Class Classification
- Hemi-Supervised Detection
- Semi-Supervised Detection/Classification
- Min-Max Classification
- Multiple Instance
- Learning with Reject
- Rare Category Detection
- Learning to Order

Additional Learning problems that can be mapped to a PLD problem

- Neyman-Pearson Classification
- Multi-class classification (with arbitrary loss)
- Multi-class extensions
  - individual sample weights
  - multi-class regression
Multi-category Prediction Problems

- **Input/Output**: Define
  - \( X = \text{input vector space} \ (e.g. \ X \subseteq \mathbb{R}^d) \)
  - \( M = \text{number of categories} \)
  - \( Y = \{Y_1, Y_2, ..., Y_M\} = \text{categorical output values} \) (not necessarily \( \{1,2,3,...\} \))
  - \( (x, y) \in X \times Y = \text{data sample} \)

- **Prediction Functions**: Define
  - \( C = \{1, ..., M\} \) the label set
  - \( f : X \rightarrow C = \text{multiclass classifier} \)
  - \( \hat{y} = Y_{f(x)} = \text{the predicted output} \)
Multi-category Prediction Problems

- **Loss Functions:**
  - General Loss: \( b(x, y, \hat{y}) = \) loss incurred at \( x \) when \( y \) is true and \( \hat{y} \) is predicted
  - 0-1 Loss: (e.g. standard multiclass classification)
    \[
    b(x, y, \hat{y}) = \begin{cases} 
    0 & y = \hat{y} \\
    1 & y \neq \hat{y} 
    \end{cases}
    \]
  - Class Dependent Loss: \( b(x, y, \hat{y}) = b(y, \hat{y}) = \) loss when \( y \) is true and \( \hat{y} \) is predicted
  - **Boosting** is an example where the loss depends on \( x \). (i.e. general loss)

- **Performance Criterion:** Let \( P \) be a probability distribution on \( X \times Y \). The average error is for the predictor \( f \) is
  \[
e(f) = E_P[b(x, y, \hat{y})] = E_P[b(x, y, \mathcal{Y}_f(x))]\]

- **Learning Problem:** Given a collection \( D_n = ((x_1, y_1), ..., (x_n, y_n)) \) of i.i.d. samples from an unknown distribution \( P \), determine a model \( \hat{f} \) whose error \( e(\hat{f}) \) is as small as possible.
1. MC: traditional $M$–class classification problem (0-1 loss)
2. MCL: traditional $M$–class classification problem with class dependent loss
3. MCGL: $M$–category prediction problem with generalized loss (generalizes the MCL criterion)

We show examples of 2,3.
1. All Pairs: (AVA) **Function Class:** A total of $M(M - 1)/2$ pairwise discriminant functions each cast a vote for one of two classes. The class with the most votes wins (needs a tie breaking scheme).

**Learning Procedure:** Design each pairwise discriminant function to discriminate between two classes (each case ignores all the other classes). (How do we modify this method for generalized loss?)
2. One vs All: (OVA) **Function Class:** A total of $M$ discriminants are computed, one for each class, and the class with the largest discriminant value wins.

**Learning Procedure:** Each of the $M$ functions is designed to discriminate between one class and all the others.

(How do we modify this method for generalized loss?)
3. **Direct:** **Function Class:** A multiclass function $f : X \rightarrow C$ similar to the OVA function is typically used.

**Learning Procedure:** The function $f$ is designed to discriminate between all classes simultaneously.

**Examples**
- decision trees
- multilayer perceptrons with backpropagation
- $M$-class GML
- $k$-nearest neighbors

**Ratchet** provides a *direct* method for *linear machines.*
Linear Machine: Each one-vs-all function is a *linear* discriminant.

\[ w_1 \cdot (1, x) \]

\[ w_2 \cdot (1, x) \]

\[ w_3 \cdot (1, x) \]

Assign to class with the largest value

\[ \omega = (w_1, w_2, \ldots, w_M) \]
How Do We Use Ratchet?

What Do We Have to Do:

1. find a data map $D \mapsto Z$ and
2. a criterion PLD $R$ on $Z$ that is calibrated to $e$
A Map $\phi$ for $M$–Class Linear Machines

Data: $D = \{(x_1, y_1, B_1), (x_2, y_2, B_2), \ldots, (x_n, y_n, B_n)\}$

where

$$B_i = \begin{bmatrix} b_{i1} \\ b_{i2} \\ \vdots \\ b_{iM} \end{bmatrix}$$

and

$$b_{ij} = \text{loss incurred at } x_i \text{ when } y_i \text{ is true and } Y_j \text{ is predicted}$$

The empirical criterion is

$$R(\omega) = \sum_{i=1}^{n} \sum_{c=1}^{M} b_{ic} I(f_\omega(x_i) = c)$$

This is an $M$-class classification error with a weight for each classification assignment of each sample.
A Map $\phi$ for $M$–Class Linear Machines

Steps in Data Map:

1. define $\xi_i = (1, x_i)$
2. then define the map $\xi_i \mapsto (..., z_{ijk}, ...) \text{ as follows}$

$$z_{ijk} = (0...0 \xi_i 0...0 -\xi_i 0...0), \quad 1 \leq j, k \leq M, j \neq k$$

Example with $M = 3$

$$\begin{pmatrix} \xi_i & -\xi_i & 0 \\ \xi_i & 0 & -\xi_i \end{pmatrix} \quad z_{i12}$$
$$\begin{pmatrix} 0 & \xi_i & -\xi_i \\ -\xi_i & \xi_i & 0 \end{pmatrix} \quad z_{i21}$$
$$\begin{pmatrix} 0 & \xi_i & -\xi_i \\ -\xi_i & 0 & \xi_i \end{pmatrix} \quad z_{i23}$$
$$\begin{pmatrix} 0 & -\xi_i & \xi_i \\ -\xi_i & 0 & \xi_i \end{pmatrix} \quad z_{i31}$$
$$\begin{pmatrix} 0 & -\xi_i & \xi_i \end{pmatrix} \quad z_{i32}$$

With $\omega = (w_1, w_2, ..., w_M)$ we have

$$(w_j \cdot \xi_i > w_k \cdot \xi_i) \iff (\omega \cdot z_{ijk} > 0)$$

Also note that $z_{ijk} = -z_{ikj}$ so

$$(\omega \cdot z_{ijk} > 0) \iff (\omega \cdot z_{ikj} < 0)$$

so only one group can be positive.
Risk Function Values in terms of the Mapped Samples

**Winner–take–all Property:**

Example with $M = 4$,

\[ \xi_i \rightarrow z_{i12}, z_{i13}, z_{i14} \quad \text{group 1} \]
\[ z_{i21}, z_{i23}, z_{i24} \quad \text{group 2} \]
\[ z_{i31}, z_{i32}, z_{i34} \quad \text{group 3} \]
\[ z_{i41}, z_{i42}, z_{i43} \quad \text{group 4} \]

- For every $\omega$ without ties, exactly 1 group is $\omega$–positive.
- For every $\omega$ with ties for the winner, 0 groups are $\omega$–positive.
- For every $\omega$ there exists an $\hat{\omega}$ with no ties such that
  - $\hat{\omega}$ gives the same winner as the *max index* tie breaking rule.
  - $Z^+(\hat{\omega}) \supseteq Z^+(\omega)$.

The risk function values:

\[
R(\omega) = \sum_{i=1}^{n} \sum_{j=1}^{M} b_{ij} I(f_\omega(x_i) = j)
\]

\[
= \sum_{i=1}^{n} \sum_{j=1}^{M} b_{ij} I(\hat{\omega} \cdot z_{ijk} > 0, \forall k \neq j)
\]

*At this point it is almost obvious that $R$ is PLD.* (proof requires a little more work)
Comments/Observations

- Each sample $\xi_i$ of dimension $d + 1$ maps to $M(M - 1)$ samples of dimension $M(d + 1)$.

- This is an extension of Kesler’s map (from 1960s) … which produced $M - 1$ samples of dimension $M(d + 1)$.

- Define
  \[ \Delta_{ijk} = b_{ik} - b_{ij}. \]
  We can prove that $R$ does not depend on the samples $z_{ijk}$ for which $\Delta_{ijk} \leq 0$, so we can discard these samples. This reduces the number of mapped samples by at least 50%.

- Furthermore, we can replace samples $z_{ijk}$ by $\Delta_{ijk}z_{ijk}$ (when $\Delta_{ijk} > 0$). With this modification Ratchet often requires fewer iterations to find a solution of equivalent value.
1. **Brute Force:** apply a nonlinear map, $z' \leftarrow \text{map}(z)$, and run Ratchet on $z'$. 

2. **Kernel Method:** replace dot products with kernel computation

$$\omega \cdot z_i \iff \sum_j \alpha_j k(z_i, z_j)$$
The Kernelized Ratchet Algorithm

**INPUT:** A "data set" \( Z = \{z_1, z_2, ..., z_n\} \).

\[ \alpha^* \leftarrow 0, \quad R^* \leftarrow R(\alpha^*) \]
\[ \alpha \leftarrow 0 \]

\{Perform the RP algorithm and track the best solution.\}

\textbf{loop}

\[ i \leftarrow \text{index chosen randomly from } \{1, 2, ..., n\} \]

\textbf{if} \( \sum_j \alpha_j k(z_i, z_j) \leq 0 \) \textbf{then}

\[ \alpha_i \leftarrow \alpha_i + 1.0 \]

\textbf{if} \( R(\alpha) < R^* \) \textbf{then}

\[ R^* \leftarrow R(\alpha) \]

\[ \alpha^* \leftarrow \alpha \]

\textbf{end if}

\textbf{end if}

\textbf{end if}

\textbf{end loop}

- This is simplest version ... (may want to pull the offset weights out of the kernel)
- Some simple speed-up options (pre-compute kernel matrix, update \( R \) instead of computing from scratch each time, etc.)
Ratchet: Strengths and Weaknesses

Advantages:
- Simplicity! (implementation is trivial, very few tuning parameters (stopping))
- Flexibility - can be applied to a large number of different learning problems
- Direct solution method that adjusts all weights simultaneously accounting for simultaneous interactions between all $M$ classes. (in contrast to pairwise training)
- Computationally Robust
  - very simple numerical operations (dot products), no divisions or matrix inverses, weights are bounded
  - robust to repeated samples (which can cause singularities in other learning algorithm)
  - trivial to kernelize
- Asymptotic Properties are very good: estimation and computation error go to zero asymptotically and with the proper choice of kernel so does the approximation error (consistency).
- Has been shown to be very effective in numerous real world problems.

Disadvantages: Ratchet is a randomized algorithm so ...
- convergence can be very slow
- different solution each time
Experiment #1

Converting Printed Documents to ASCII:

**Basic OCR System**

- printed document (PD) → scanner → digitized document (DD) → convert to ASCII → ASCII document (AD)

**Enhanced OCR System**

- PD → scanner → DD → restore using 1 of M → RDD → convert to ASCII → AD
  - c → feature map (FM) → classifier → human makes comparison and determines errors → y
I feel that it is both necessary to be undertaken without delay. Conclusions and recommendations by the Commission will give whatever those parts of the program are qualified.
Experiment #1

- **Performance Criterion:**
  - weighted multiclass with a *different cost for each class assignment for each sample*
  - non-zero cost even when the "best" class is chosen

- **Results:**
  1. A simple synthetic experiment
  2. Real Corpus data
Experiment #1

- **synthetic data:**
  - for each $x_i$ there is an $M$-vector $(y_{i1}, y_{i2}, \ldots, y_{iM})$ of synthetic error rates,
  - each component $y_{ij}$ is determined by a simple parametric model $p(y_j|x)$
- $M = 3$, $d = 2$ and $d = 5$
- training set sizes vary from $n = 25$ to 400.
- test set size is 10,000
- optimal solution illustrated on empirical distribution below
Classifier Design Methods

**MV01:** (AVA) An “all pairs” classifier trained with the Pocket algorithm.

**kNN:** The $k$–nearest neighbor method with Euclidean metric.

**LM:** An $M$–class linear machine trained with GMR to minimize the empirical OCR error.

**LM01:** An $M$–class linear machine trained with GMR to minimize “classification error”.

**LM01PAIR:** (OVA) An $M$–class linear machine trained with Pocket to minimize “pairwise classification error”.

**QM:** An $M$–class quadratic machine trained with GMR to minimize the empirical OCR error.

**QM01:** An $M$–class quadratic machine trained with GMR to minimize the same classification error as LM01 above.

**QM01PAIR:** (OVA) An $M$–class quadratic machine trained with Pocket to minimize the same classification error as LM01PAIR above.
1. optimal
2. training set
3. kNN (k=10)
4. LM
5. QM
6. LM01
7. QM01
Two-Dimensional Experiments

Two-dimensional experiments: kNN and Linear Machines

Two-dimensional experiments: kNN and Quadratic Machines
Five-Dimensional Experiments

Five-dimensional experiments: kNN and Linear Machines

Five-dimensional experiments: kNN and Quadratic Machines
Real Corpus Data

- **Data:**
  - 1445 documents from a real world corpus
  - $d = 7$ quality measures (features input to the classifier)
  - $M = 10$ classes (restoration techniques, 9 + original)
Experiment #1

Summary of Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower Bound</td>
<td>0.0810</td>
</tr>
<tr>
<td>No Restoration</td>
<td>0.1105</td>
</tr>
<tr>
<td>MV01</td>
<td>0.1018</td>
</tr>
<tr>
<td>kNN</td>
<td>0.0983</td>
</tr>
<tr>
<td>LM</td>
<td>0.0977</td>
</tr>
<tr>
<td>QM</td>
<td>0.0988</td>
</tr>
<tr>
<td>LM01</td>
<td>0.1019</td>
</tr>
<tr>
<td>QM01</td>
<td>0.1002</td>
</tr>
<tr>
<td>LM01PAIR</td>
<td>0.1036</td>
</tr>
<tr>
<td>QM01PAIR</td>
<td>0.1028</td>
</tr>
</tbody>
</table>

- MV01 (previous AVA method) is an improvement over “no restoration”
- kNN, LM and QM all provide improvement over previous method (MV01)
- LM has best error estimate
- optimizing OCR error rate (LM and QM) works better than optimizing classification error (LM01 and QM01)
- optimizing overall error (LM01 and QM01) works better than pairwise optimization
Multiclass Regression

Model: A model implements a function $f : X \to Y$ where $Y$ is a finite set with $M$ discrete ordered values.

Goal: Design an $M$-class classifier where the classes are ordered.
Multiclass Regression

- **Loss Function for MC Regression:** (define $0^0 = 0$)
  
  $$b(x, y, \hat{y}) = |y - \hat{y}|^p = |y - \mathcal{Y}_{f(x)}|^p$$

  where
  
  - $p = 2$ ⇒ mean squared error
  - $p = 1$ ⇒ mean absolute error
  - $p = 0$ ⇒ mean counting error (traditional multiclass error)

- **Performance Criterion:**
  
  $$e(f) = E_P \left[ |y - \mathcal{Y}_{f(x)}|^p \right] \approx \frac{1}{n} \sum_{i=1}^{n} |y_i - \mathcal{Y}_{f(x_i)}|^p$$

- **Extension:** It is easy to extend to the $x$-dependent loss
  
  $$b(x, y, \hat{y}) = a(x) |y - \hat{y}|^p$$
Experiment #2

High Speed Video  \( \rightarrow \)  Predictor  \( \rightarrow \)  Hardness

\( (4000 \text{ f/s}) \)

Ground Truth Data

<table>
<thead>
<tr>
<th>1010 C</th>
<th>1177 C</th>
</tr>
</thead>
<tbody>
<tr>
<td>482 C</td>
<td>42.3–43.2</td>
</tr>
<tr>
<td>565 C</td>
<td>35.9–36.2</td>
</tr>
<tr>
<td>621 C</td>
<td>32.6–33.2</td>
</tr>
</tbody>
</table>
Experiment #2

- We cannot measure the hardness value at every point, but we can measure a hardness value interval.
- Our goal is to predict the hardness value interval:

```
| H33 | H35 | H36 | H39 | H43 | H45 |
```

Performance Criterion:
- The average absolute difference between the mid value of the predicted interval and the mid value of the true interval.
- weighted multiclass with a different cost for each class assignment for each sample
- classes are ordered

Cost Matrix

<table>
<thead>
<tr>
<th></th>
<th>H45</th>
<th>H43</th>
<th>H39</th>
<th>H36</th>
<th>H35</th>
<th>H33</th>
</tr>
</thead>
<tbody>
<tr>
<td>H45</td>
<td>0</td>
<td>2.6</td>
<td>6.7</td>
<td>9.4</td>
<td>10.7</td>
<td>12.6</td>
</tr>
<tr>
<td>H43</td>
<td>2.6</td>
<td>0</td>
<td>4.1</td>
<td>6.8</td>
<td>8.1</td>
<td>10.0</td>
</tr>
<tr>
<td>H39</td>
<td>6.7</td>
<td>4.1</td>
<td>0</td>
<td>2.7</td>
<td>4.0</td>
<td>5.9</td>
</tr>
<tr>
<td>H36</td>
<td>9.4</td>
<td>6.8</td>
<td>2.7</td>
<td>0</td>
<td>1.3</td>
<td>3.2</td>
</tr>
<tr>
<td>H35</td>
<td>10.7</td>
<td>8.1</td>
<td>4.0</td>
<td>1.3</td>
<td>0</td>
<td>1.9</td>
</tr>
<tr>
<td>H33</td>
<td>12.6</td>
<td>10.0</td>
<td>5.9</td>
<td>3.2</td>
<td>1.9</td>
<td>0</td>
</tr>
</tbody>
</table>
Experiment #2

$d = 8$ features, several thousand data samples

Results:

<table>
<thead>
<tr>
<th>Method</th>
<th>Weighted Classification Error (0,10.3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive</td>
<td>3.9</td>
</tr>
<tr>
<td>Ratchet</td>
<td>2.16</td>
</tr>
<tr>
<td>kNN</td>
<td>1.95</td>
</tr>
</tbody>
</table>
Experiment #3

- **Goal:** Predict the number of semesters it takes a student to graduate.
- **Features:** (from Tushar)
  
  | SEMGPA01 | SEMGPA02 | SEMGPA03 |
  | SEMGPA04 | SEMGPA05 | SEMGPA06 |
  | CUMHRS02 | CUMHRS04 | CUMHRS06 |

- **Data:**
  - UNM undergraduate students who started in 2006-2010
  - kept only students with $> 6$ semesters and no missing feature values (9962)
  - $M = 13$ "classes" (7,8,...,19)

- **Train/Test:** 80%/20% (one hold-out set)

- **Methods:**
  - Standard Regression: Linear and Nonlinear (spline model) (Reg $p = 2$)
  - Multiclass Regression: Linear (and Quadratic) Machine with Ratchet (MReg $p = 1$ and Reg $p = 2$)
  - Multiclass Regression: kNN method (MReg $p = 1$)
  - Multiclass Classification: CART (MCC $p = 0$)
  - Multiclass Classification: Linear (and Quadratic) Machine with Ratchet (MCC $p = 0$)
Experiment #3

The graph shows the distribution of the number of students across different numbers of semesters. The x-axis represents the number of semesters, ranging from 6 to 20, and the y-axis represents the number of students, ranging from 0 to 3000. The bars indicate the number of students for each semester range.
**Experiment #3**

_Naive Method:_ shown in ()

<table>
<thead>
<tr>
<th>Method</th>
<th>Classification Error Rate</th>
<th>Mean Absolute Error</th>
<th>Mean Squared Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reg-Lin</td>
<td>.73 (.74)</td>
<td>1.27 (1.44)</td>
<td>3.1 (3.4)</td>
</tr>
<tr>
<td>Reg-NonLin</td>
<td>.73 (.74)</td>
<td>1.24 (1.44)</td>
<td>3.0 (3.4)</td>
</tr>
<tr>
<td>Reg-Ratchet-Lin</td>
<td>.73 (.74)</td>
<td></td>
<td>3.26 (3.57*)</td>
</tr>
<tr>
<td>MCRReg-Ratchet-Lin</td>
<td>.72 (.74)</td>
<td>1.31 (1.4)</td>
<td>3.26 (3.57*)</td>
</tr>
<tr>
<td>MCRReg-Ratchet-Quad</td>
<td>.7 (.74)</td>
<td>1.26 (1.4)</td>
<td></td>
</tr>
<tr>
<td>MCRReg-kNN</td>
<td>.68 (.74)</td>
<td>1.2 (1.4)</td>
<td></td>
</tr>
<tr>
<td>MCC-CART</td>
<td>.65 (.74)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCC-Ratchet-Lin</td>
<td>.67 (.74)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCC-Ratchet-Quad</td>
<td>.68 (.74)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCC-kNN</td>
<td>.68 (.74)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Naive method is different for linear vs multiclass regression (\(\bar{y}\) vs most common category)
### Linear Regression - MCC Matrix

<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0</td>
<td>112</td>
<td>66</td>
<td>403</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>295</td>
<td>1288</td>
<td>964</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>63</td>
<td>773</td>
<td>1329</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>56</td>
<td>602</td>
<td>1561</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>9</td>
<td>99</td>
<td>712</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>9</td>
<td>118</td>
<td>797</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>1</td>
<td>22</td>
<td>241</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>2</td>
<td>33</td>
<td>216</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>83</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>60</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>---</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>7</td>
<td>26</td>
<td>6</td>
<td>193</td>
<td>354</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>13</td>
<td>621</td>
<td>1176</td>
<td>737</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td>259</td>
<td>933</td>
<td>960</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>246</td>
<td>772</td>
<td>1192</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>35</td>
<td>187</td>
<td>597</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>44</td>
<td>198</td>
<td>682</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>6</td>
<td>40</td>
<td>215</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>5</td>
<td>47</td>
<td>198</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>0</td>
<td>11</td>
<td>74</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>4</td>
<td>13</td>
<td>48</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>7</td>
<td>246</td>
<td>99</td>
<td>77</td>
<td>105</td>
<td>30</td>
<td>23</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>124</td>
<td>1510</td>
<td>522</td>
<td>337</td>
<td>24</td>
<td>28</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>93</td>
<td>791</td>
<td>683</td>
<td>507</td>
<td>37</td>
<td>52</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>64</td>
<td>698</td>
<td>634</td>
<td>677</td>
<td>62</td>
<td>83</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>38</td>
<td>153</td>
<td>205</td>
<td>332</td>
<td>28</td>
<td>63</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>42</td>
<td>149</td>
<td>249</td>
<td>380</td>
<td>49</td>
<td>53</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>19</td>
<td>36</td>
<td>55</td>
<td>113</td>
<td>19</td>
<td>22</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>34</td>
<td>68</td>
<td>109</td>
<td>13</td>
<td>13</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>6</td>
<td>10</td>
<td>15</td>
<td>45</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>16</td>
<td>8</td>
<td>23</td>
<td>9</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
## MC-CART - MCC Matrix

<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>315</td>
<td>58</td>
<td>22</td>
<td>186</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>148</td>
<td>1548</td>
<td>378</td>
<td>473</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>125</td>
<td>733</td>
<td>532</td>
<td>775</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>132</td>
<td>601</td>
<td>383</td>
<td>1103</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>67</td>
<td>101</td>
<td>103</td>
<td>549</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>54</td>
<td>145</td>
<td>105</td>
<td>620</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>30</td>
<td>24</td>
<td>26</td>
<td>184</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>17</td>
<td>40</td>
<td>23</td>
<td>171</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>8</td>
<td>11</td>
<td>4</td>
<td>64</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>6</td>
<td>11</td>
<td>6</td>
<td>42</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>7</td>
<td>263</td>
<td>50</td>
<td>0</td>
<td>252</td>
<td>0</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>108</td>
<td>1607</td>
<td>0</td>
<td>815</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>93</td>
<td>934</td>
<td>0</td>
<td>1107</td>
<td>1</td>
<td>1</td>
<td>17</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>96</td>
<td>770</td>
<td>0</td>
<td>1320</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>9</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>46</td>
<td>146</td>
<td>0</td>
<td>609</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>48</td>
<td>192</td>
<td>0</td>
<td>664</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>6</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>27</td>
<td>31</td>
<td>0</td>
<td>198</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>19</td>
<td>49</td>
<td>0</td>
<td>176</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>7</td>
<td>6</td>
<td>0</td>
<td>69</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
<td>13</td>
<td>0</td>
<td>47</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
THANK YOU!