# Large-Scale Nearest Neighbor Classification with Statistical Guarantee


Jiexin Duan

Department of Statistics

Purdue University

Joint Work with Guang Cheng and Xingye Qiao


Mar 13, 2018

University of Wisconsin-Madison

# The Big Data Era

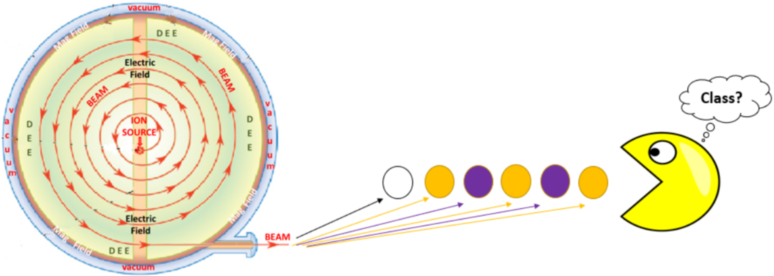
*"There were 5 exabytes of information created between the dawn of civilization through 2003, but that much information is now created every 2 days."*

–Eric Schmidt, Google CEO (2001-2011)

- **Volume**
- Variety
- Velocity

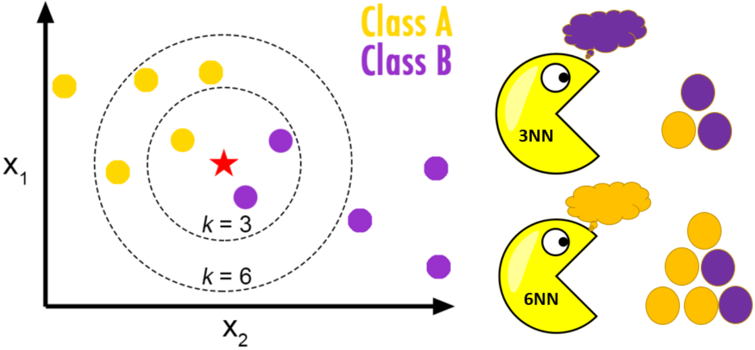
# SUSY Data Set

- Size: 5,000,000 particles from the accelerator
- Predictor variables: 18 (properties of the particles)
- Goal: distinguish the two classes of a signal process
- Data source: UCI Machine Learning Repository



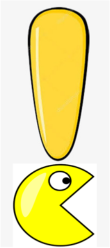
# Nearest Neighbor Classification

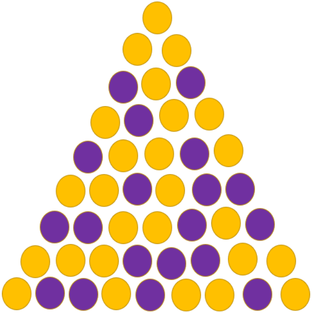
The $k$NN classifier predicts the class of $x \in \mathbb{R}^d$ to be the most frequent class of its $k$ nearest neighbors (Euclidean distance).



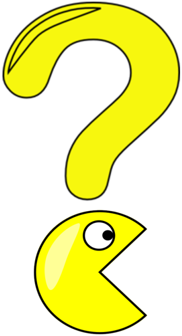
# Computational Challenges in Big Data for $k$NN Classifiers

- Time complexity: $O(dN + kN)$
  - $dN$: computing distances from the query point to all $N$ observations in $\mathbb{R}^d$
  - $kN$: selecting the $k$ nearest distances out of the $N$ distances
- Space complexity: $O(dN)$
- Here, $N$ is the size of the entire dataset





# How to Conquer "Big Data?"



Train the total data at one time: oracle $k$NN

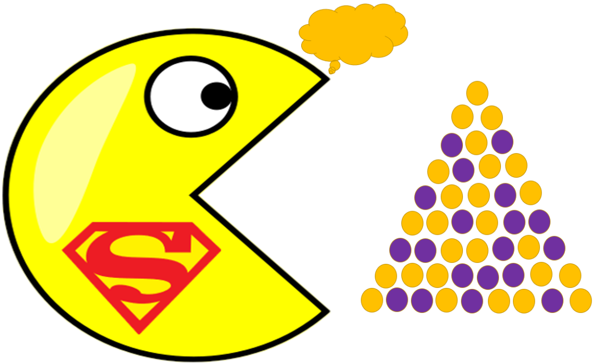
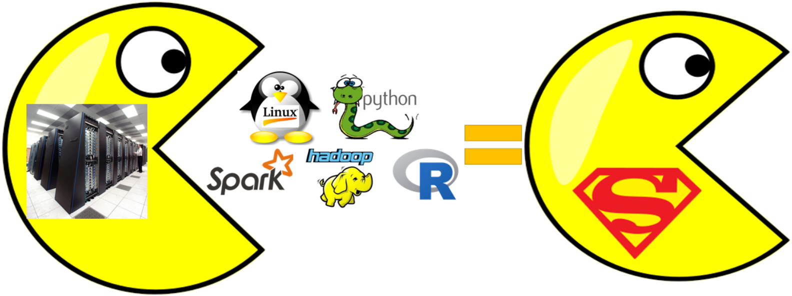


Figure: I'm Pac-Superman!

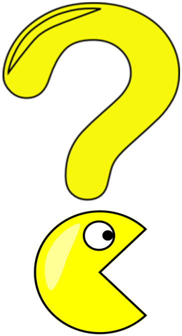
# Construction of Pac-Superman



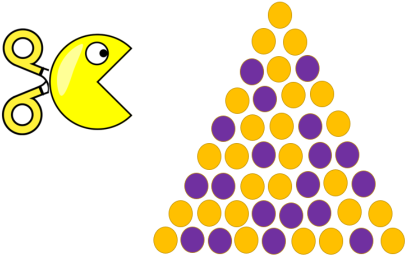

To build a Pac-superman for oracle-$k$NN, we need:

- Expensive Super-computer
- Operation system, e.g. Linux
- Statistical software designed for big data, e.g. Spark, Hadoop
- Write complicated algorithm (eg. MapReduce) which is hard to extend to other classifier

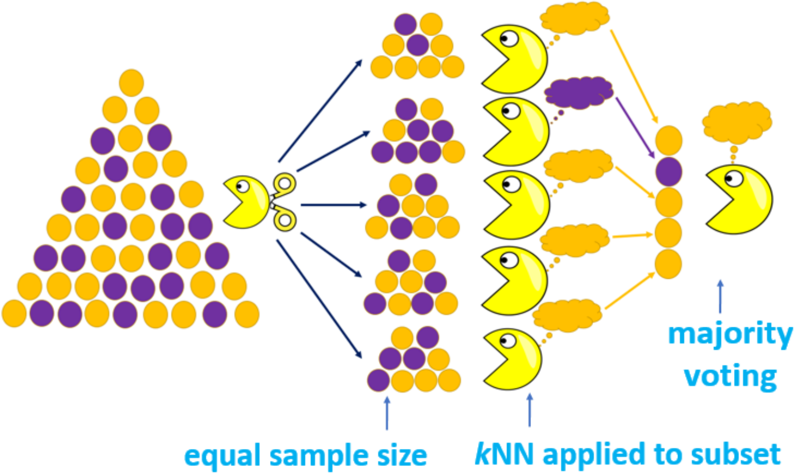
What can we do?



What about splitting the data?



# Divide & Conquer (D&C) Framework



# Big-$k$NN for SUSY Data

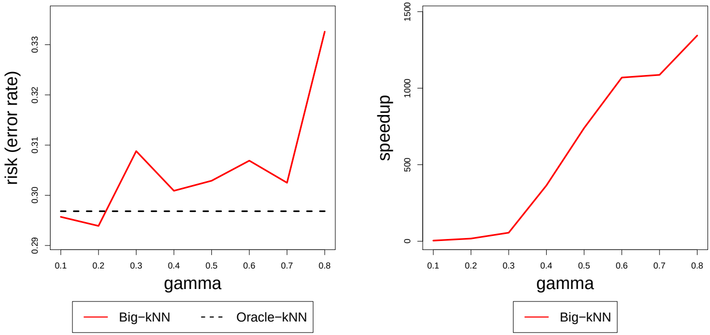
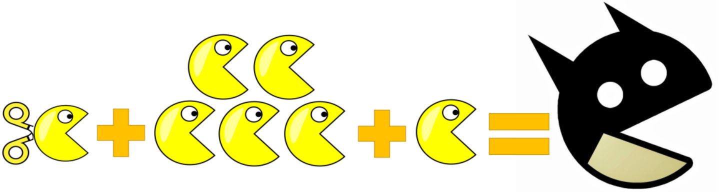



Figure: Risk and Speedup for Big-kNN.

- Risk of classifier $\phi$: $R(\phi) = \mathbb{P}(\phi(X) \neq Y)$
- Oracle-$k$NN: $k$NN trained by the total dataset
- Speedup: running time ratio between Oracle-$k$NN & Big-$k$NN
- Number of subsets: $s = N^{\gamma}$, $\gamma = 0.1, 0.2, \ldots 0.8$

# Construction of Pac-**Bat**man (**B**ig D**a**t**a**)



# Weighted Nearest Neighbor Classifiers (WNN)

**Definition: Weighted Nearest Neighbor Classifier (WNN)**

In a dataset with size n, the WNN classifier has weight $w_{ni}$ on the $i$-th neighbor of x:

$$\widehat{\phi}_n^{w_n}(x) = \mathbb{1}\{\sum_{i=1}^{n} w_{ni} \mathbb{1}\{Y_{(i)} = 1\} \geq \frac{1}{2}\} \; s.t. \; \sum_{i=1}^{n} w_{ni} = 1$$

When $w_{ni} = k^{-1}\mathbb{1}\{1 \leq i \leq k\}$, WNN reduces to $k$NN.

# Big Weighted Nearest Neighbor Classifiers (BigWNN)

**Definition: Big Weighted Nearest Neighbor Classifier (BigWNN)**

In a big data set with size $N = sn$, denote $\widehat{\phi}_n^{(j)}(x)$ as the local WNN in $j$-th subset with size n. For any subset, the weights $w_{ni}$ are the same. Then the BigNN is constructed as:

$$\widehat{\phi}_{n,s}^{Big}(x) = \mathbb{1}\{s^{-1}\sum_{j=1}^{s}\widehat{\phi}_n^{(j)}(x) > 1/2\}$$

## Statistical Guarantees

- Accuracy
  - Regret=Expected Risk−Bayes Risk= $\mathbb{E}_{\mathcal{D}}\left[R(\widehat{\phi}_n)\right] - R(\phi^{\mathrm{Bayes}})$
  - A small Regret represents an accurate classifier
- Stability

**Definition: Classification Instability (CIS)**

Define classification instability of a classification procedure $\Psi$ as

$$\mathrm{CIS}(\Psi) = \mathbb{E}_{\mathcal{D}_1,\mathcal{D}_2}\left[\mathbb{P}_X\left(\widehat{\phi}_{n1}(X) \neq \widehat{\phi}_{n2}(X)\right)\right],$$

where $\widehat{\phi}_{n1}$ and $\widehat{\phi}_{n2}$ are the classifiers obtained by applying the classification procedure $\Psi$ to $\mathcal{D}_1$ and $\mathcal{D}_2$ which are i.i.d. copies of $\mathcal{D}$.

  - A small CIS represents a stable classifier

# Asymptotic Regret of BigWNN

**Theorem**

*Under regularity assumptions, with $s$ upper bounded by subset size $n$, as $n, s \to \infty$, we have*

$$\mathit{Regret}(\mathit{BigWNN}) \approx B_1 s^{-1} \sum_{i=1}^{n} w_{ni}^2 + B_2 \left( \sum_{i=1}^{n} \frac{\alpha_i w_{ni}}{n^{2/d}} \right)^2,$$

*where $\alpha_i = i^{1+\frac{2}{d}} - (i-1)^{1+\frac{2}{d}}$, $w_{ni}$ are the local weights, constants $B_1$ and $B_2$ are based on the underlying distribution.*

# Asymptotic Regret of BigWNN

**Theorem**

*Under regularity assumptions, with $s$ upper bounded by subset size $n$, as $n, s \to \infty$, we have*

$$\mathit{Regret}(\mathit{BigWNN}) \sim B_1 s^{-1} \sum_{i=1}^{n} w_{ni}^2 + B_2 \left( \sum_{i=1}^{n} \frac{\alpha_i w_{ni}}{n^{2/d}} \right)^2,$$

*where $\alpha_i = i^{1+\frac{2}{d}} - (i-1)^{1+\frac{2}{d}}$, $w_{ni}$ are the local weights, constants $B_1$ and $B_2$ are based on the underlying distribution.*

*Variance part*

# Asymptotic Regret of BigWNN

**Theorem**

*Under regularity assumptions, with s upper bounded by subset size n, as $n, s \to \infty$, we have*

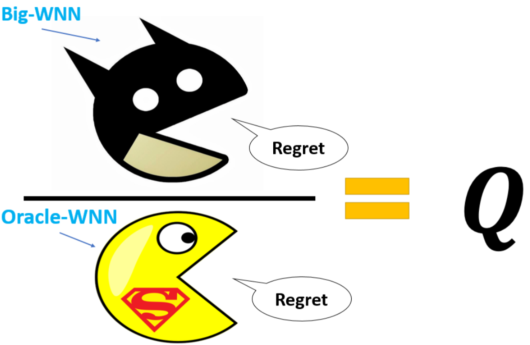
$$Regret(BigWNN) \sim B_1 s^{-1} \sum_{i=1}^{n} w_{ni}^2 + B_2 \left( \sum_{i=1}^{n} \frac{\alpha_i w_{ni}}{n^{2/d}} \right)^2,$$

*where $\alpha_i = i^{1+\frac{2}{d}} - (i-1)^{1+\frac{2}{d}}$, $w_{ni}$ are the local weights, constants $B_1$ and $B_2$ are based on the underlying distribution.*

*Variance part*

*Bias part*

# Asymptotic Regret Comparison




**Round 1**

# Asymptotic Regret Comparison

**Theorem**

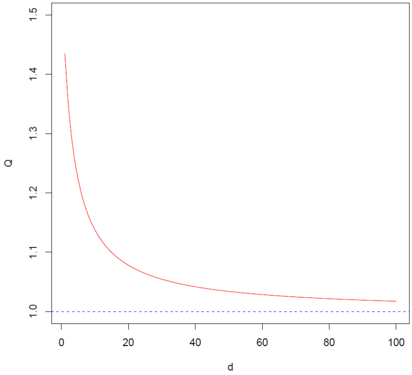
*Under regularity assumptions, as $n, s \to \infty$, given an OracleWNN classifier, we can design BigWNN by adjusting its weights according to those in the oracle version (e.g. in Big-kNN case, given oracle $k^O$, setting $k = \lfloor (\frac{\pi}{2})^{\frac{d}{d+4}} \frac{k^O}{s} \rfloor$) s.t.:*

$$\frac{Regret(BigWNN)}{Regret(OracleWNN)} \to Q$$

*where $Q = (\frac{\pi}{2})^{\frac{4}{d+4}}$.*

- We name $Q$ the **Majority Voting Quotient (MVQ)**.

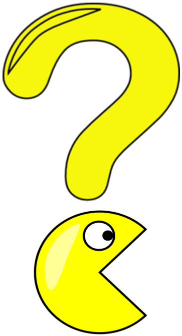
## MVQ converges to one as $d$ grows



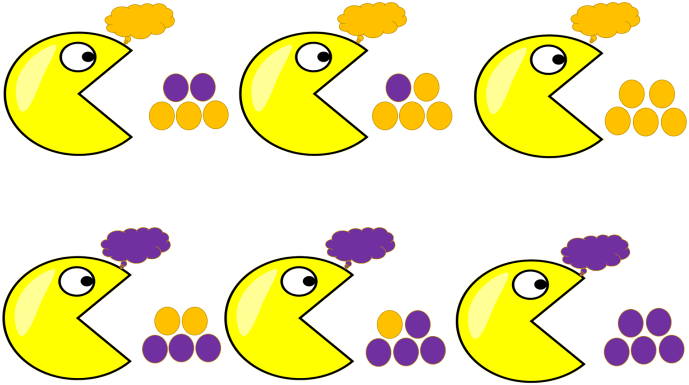

For example:

- d=1, Q=1.44
- d=2, Q=1.35
- d=5, Q=1.22
- d=10, Q=1.14
- d=20, Q=1.08
- d=50, Q=1.03
- d=100, Q=1.02

# Why there is a constant MVQ?



Accuracy loss during the transformation from (continuous) percentage to (discrete) 0-1 label



Only one majority voting in oracle classifier

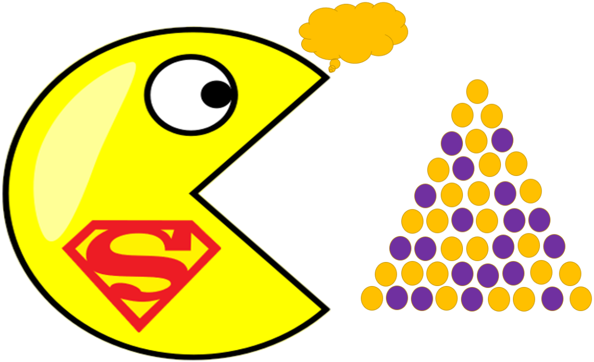
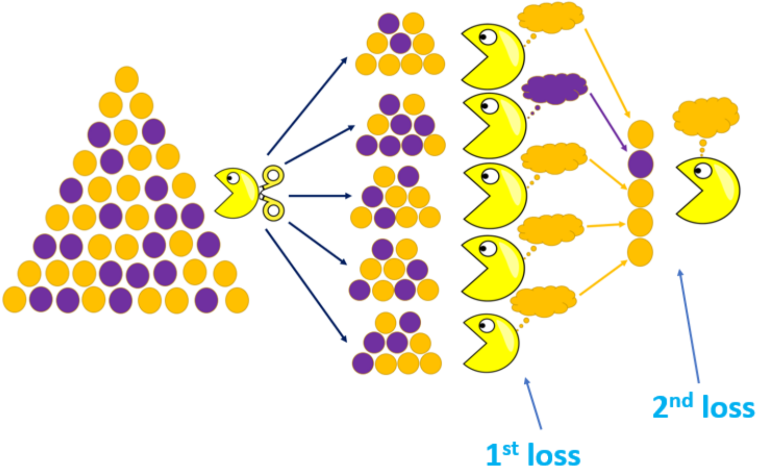


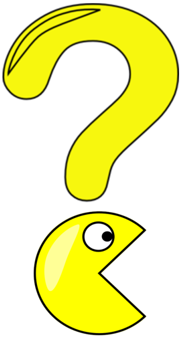
Figure: I'm Pac-Superman!

# Majority Voting/Accuracy Loss Twice in D&C Framework

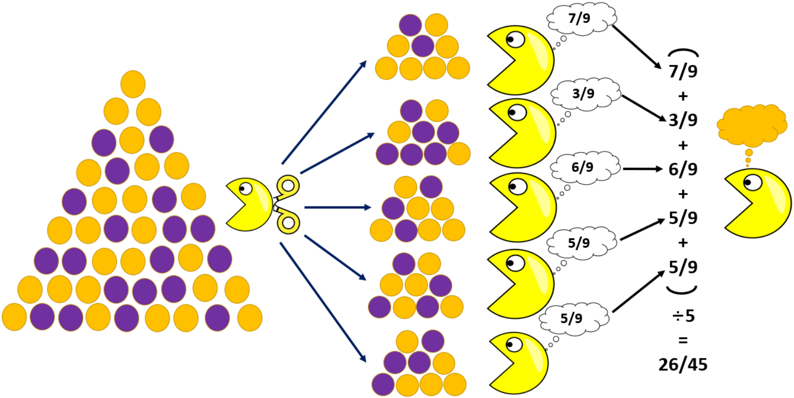


# D&C with Majority Voting Once

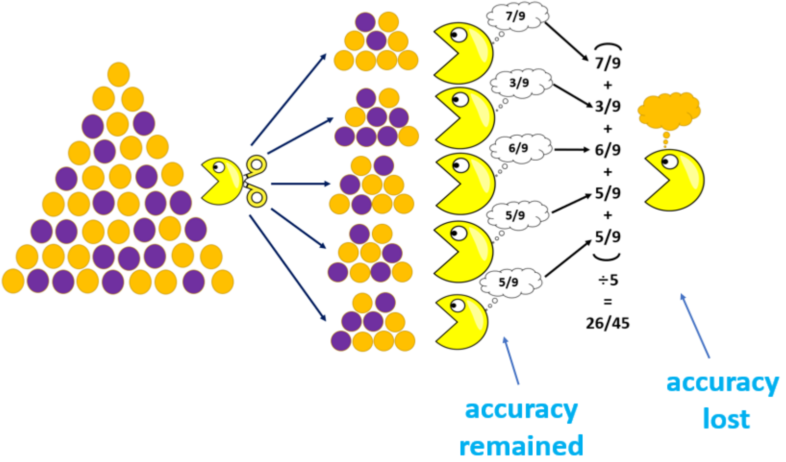
Is it possible to apply majority voting once in D&C?



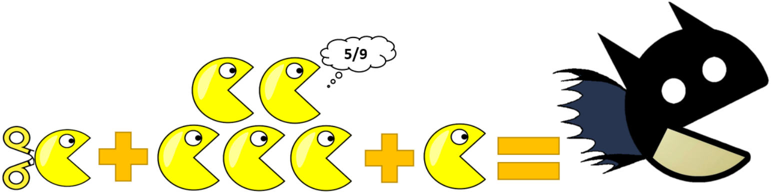
# Divide & Conquer (D&C) Framework–Continuous Version



# Majority Voting/Accuracy Loss Once in D&C Framework–Continuous Version



# Construction of Pac-**Bat**man (Continuous Version)



# BigWNN (Continuous Version) (C-BigWNN)

**Definition: Continuous Weighted Big Nearest Neighbor Classifier (C-BigWNN)**

In a big data set with size $N = sn$, denote $\widehat{\phi}_n^{(j)}(x)$ as the nearest neighbor classifier in $j$-th subset with size n. For any subset, the weights $w_{ni}$ are the same. Then the C-BigNN is constructed as:

$$\widehat{\phi}_{n,s}^{CBig}(x) = \mathbb{1}\{\frac{1}{s}\sum_{j=1}^{s}\sum_{i=1}^{n} w_{ni,j} Y_{(i),j}(x) > 1/2\}$$

# C-Big-$k$NN for SUSY Data

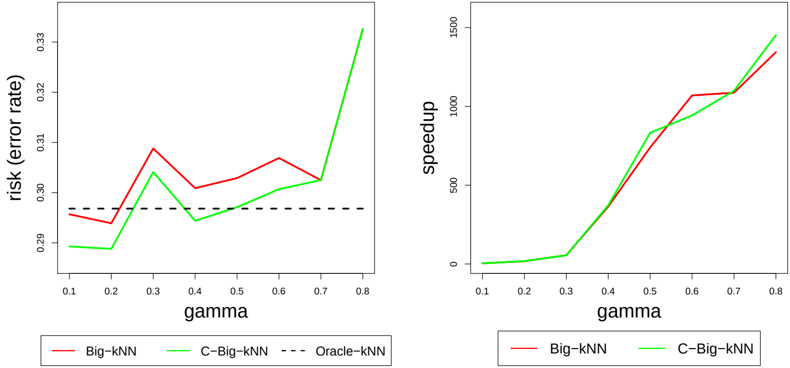
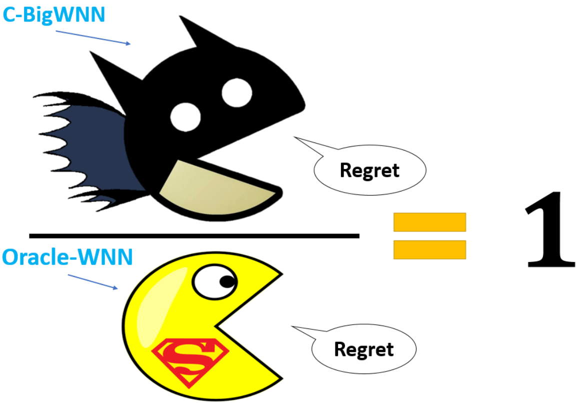



Figure: Risk and Speedup for Big-$k$NN and C-Big-$k$NN.

# Asymptotic Regret Comparison




**Round 2**

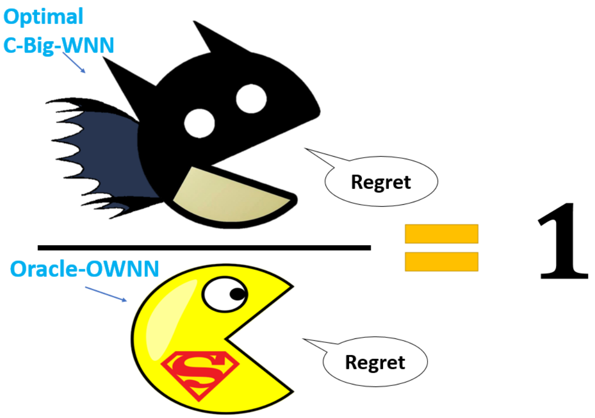
# Asymptotic Regret Comparison

**Theorem**

Under regularity assumptions, as $n, s \to \infty$, given an OracleWNN classifier, we can design C-BigWNN by adjusting its weights according to those in the oracle version (e.g. in C-Big-$k$NN case, given oracle $k^O$, setting $k = \lfloor \frac{k^O}{s} \rfloor$) s.t.

$$\frac{\mathrm{Regret(C\text{-}BigWNN)}}{\mathrm{Regret(OracleWNN)}} \to 1$$

- **Remark:** $k = \lfloor \frac{k^O}{s} \rfloor$ is different from the discrete version




# Round 3

# Optimal C-Big-WNN Classifier

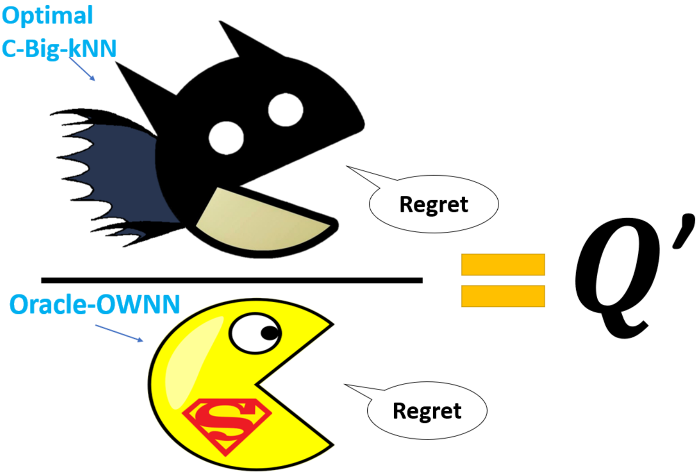
**Theorem**

*Under regularity assumptions, as $n, s \to \infty$, we can design Optimal C-Big-WNN by adjusting its local weights s.t.*

$$\frac{Regret(Optimal\ C\text{-}Big\text{-}WNN)}{Regret(Oracle\text{-}OWNN)} \to 1$$

- Oracle-OWNN is the optimal weighted nearest neighbor classified (OWNN) trained using the entire dataset.
- optimal weighted nearest neighbor classified (OWNN) is defined by Samworth (2012), and it minimizes the asymptotic regret of WNN.

# Asymptotic Regret Comparison




**Round 4**

# Optimal C-Big-$k$NN Classifier

**Theorem**

*Under regularity assumptions, as $n, s \to \infty$, we can design Optimal C-Big-kNN by adjusting its weights* $k^{opt} = \lfloor \frac{k^{O,opt}}{s} (\frac{\pi}{2})^{\frac{d}{d+4}} \rfloor$ *s.t.*

$$\frac{Regret(Optimal\ C\text{-}Big\text{-}kNN)}{Regret(Oracle\text{-}OWNN)} \to Q'$$

*where* $Q' = 4^{d/d+4} (\frac{d+4}{2d+4})^{(2d+4)/(d+4)}$ *and* $1 < Q' < 2$

# Asymptotic CIS Comparison

## Theorem

*Under regularity assumptions, as $n, s \to \infty$, we can design the Optimal C-BigWNN by adjusting its weights the same way as the regret theorem s.t.*

$$\frac{CIS(Optimal\ C\text{-}BigWNN)}{CIS(Oracle\text{-}OWNN)} \to 1$$

## Corollary

*Under regularity assumptions, as $n, s \to \infty$, we can design the Optimal Big-$k$NN by setting its subset weights the same way as the regret theorem s.t.*

$$\frac{CIS(Optimal\ C\text{-}Big\text{-}kNN)}{CIS(Oracle\text{-}OWNN)} \to \sqrt{Q'}$$

# Simulation Analysis–Setup

Consider the classification problem for Big-$k$NN and C-Big-$k$NN:

- Sample size: $N = 27,000$
- Dimensions: $d = 4, 6, 8$
- $P_0 \sim N(0_d, \mathbb{I}_d)$ and $P_1 \sim N(\frac{2}{\sqrt{d}} 1_d, \mathbb{I}_d)$
- Prior class probability: $\pi_1 = Pr(Y = 1) = 1/3$
- Number of neighbors in Oracle-$k$NN: $k^O = N^{0.7}$
- Number of subsamples in D&C: $s = N^\gamma$, $\gamma = 0.1, 0.2, \ldots 0.8$
- Number of neighbors in Big-$k$NN: $k^d = \lfloor (\frac{\pi}{2})^{\frac{d}{d+4}} \frac{k^O}{s} \rfloor$
- Number of neighbors in C-Big-$k$NN: $k^c = \lfloor \frac{k^O}{s} \rfloor$

# Simulation Analysis–Empirical Risk

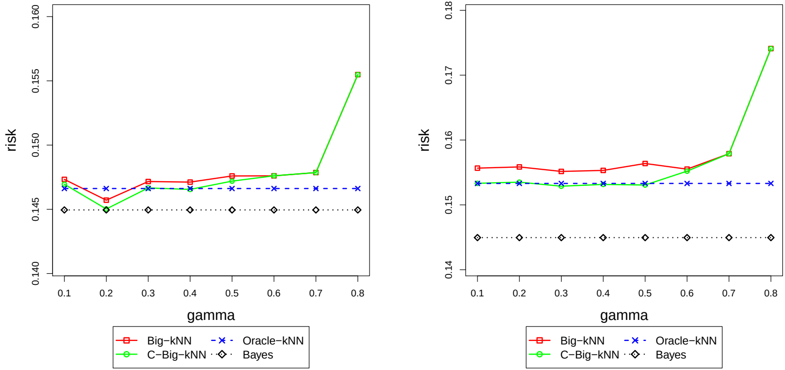



Figure: Empirical Risk (Testing Error). Left/Right: $d = 4/8$.

# Simulation Analysis–Running Time

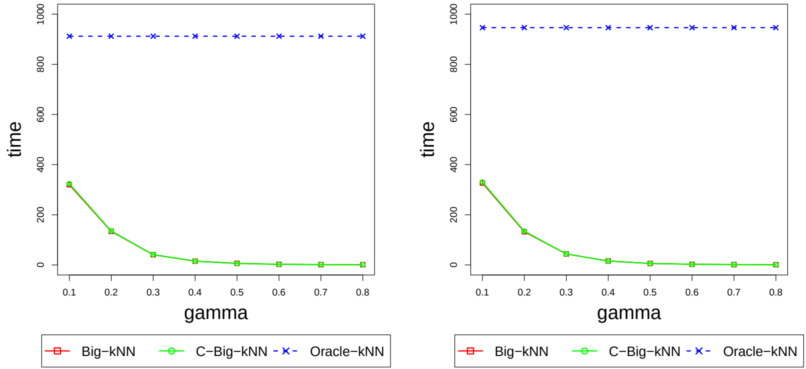



Figure: Running Time. Left/Right: $d = 4/8$.

# Simulation Analysis–Empirical Regret Ratio

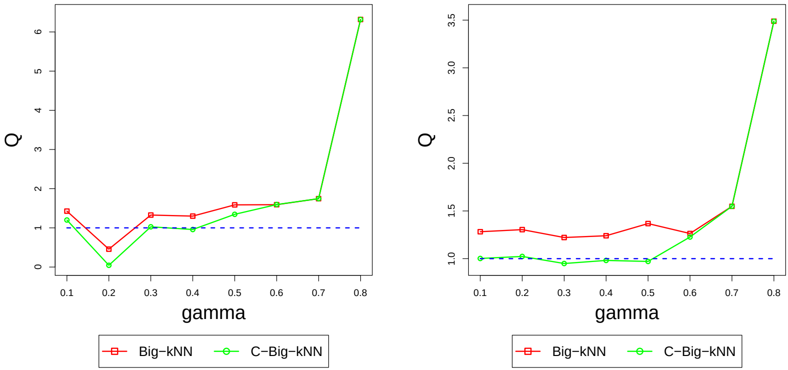



Figure: Empirical Ratio of Regret. Left/Right: $d = 4/8$.

- Q: Regret(Big-$k$NN)/Regret(Oracle-$k$NN) or Regret(C-Big-$k$NN)/Regret(Oracle-$k$NN)

# Simulation Analysis–Empirical CIS

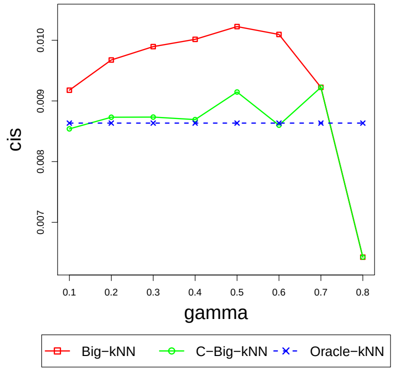
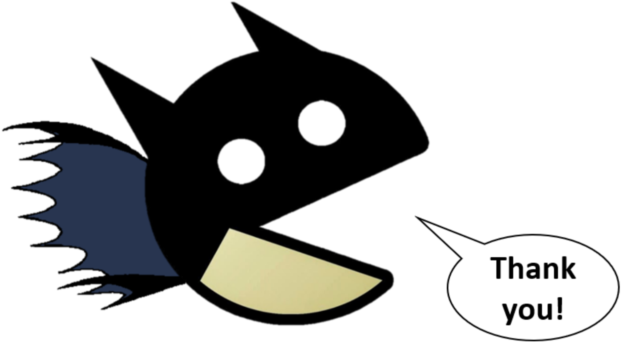



Figure: Empirical CIS. Left/Right: $d = 4/8$.

# Real Data Analysis

| Data | Size | Dim | Big-$k$NN | C-Big-$k$NN | Oracle-$k$NN | Speedup |
|---|---|---|---|---|---|---|
| htru2 | 17898 | 8 | 3.72 | 3.36 | **3.34** | 21.27 |
| gisette | 6000 | 5000 | 12.86 | **10.77** | 10.78 | 12.83 |
| musk1 | 476 | 166 | 36.43 | **33.87** | 35.71 | 3.6 |
| musk2 | 6598 | 166 | 10.43 | **9.98** | 10.14 | 14.68 |
| occup | 20560 | 6 | 5.09 | **4.87** | 5.19 | 20.31 |
| credit | 30000 | 24 | 21.85 | **21.37** | 21.5 | 22.44 |
| SUSY | 5000000 | 18 | 30.66 | 30.08 | **30.01** | 68.45 |

Table: Test error (Risk): Big-$k$NN compared to oracle-$k$NN in real datasets. Best performance is shown in bold-face. The speedup factor is defined as computing time of oracle-$k$NN divided by the time of the slower Big-$k$NN method. Oracle $k = N^{0.7}$, number of subsets $s = N^{0.3}$.


Thank you!