Large-Scale Nearest Neighbor Classification with Statistical Guarantees

Guang Cheng

Big Data Theory Lab
Department of Statistics
Purdue University

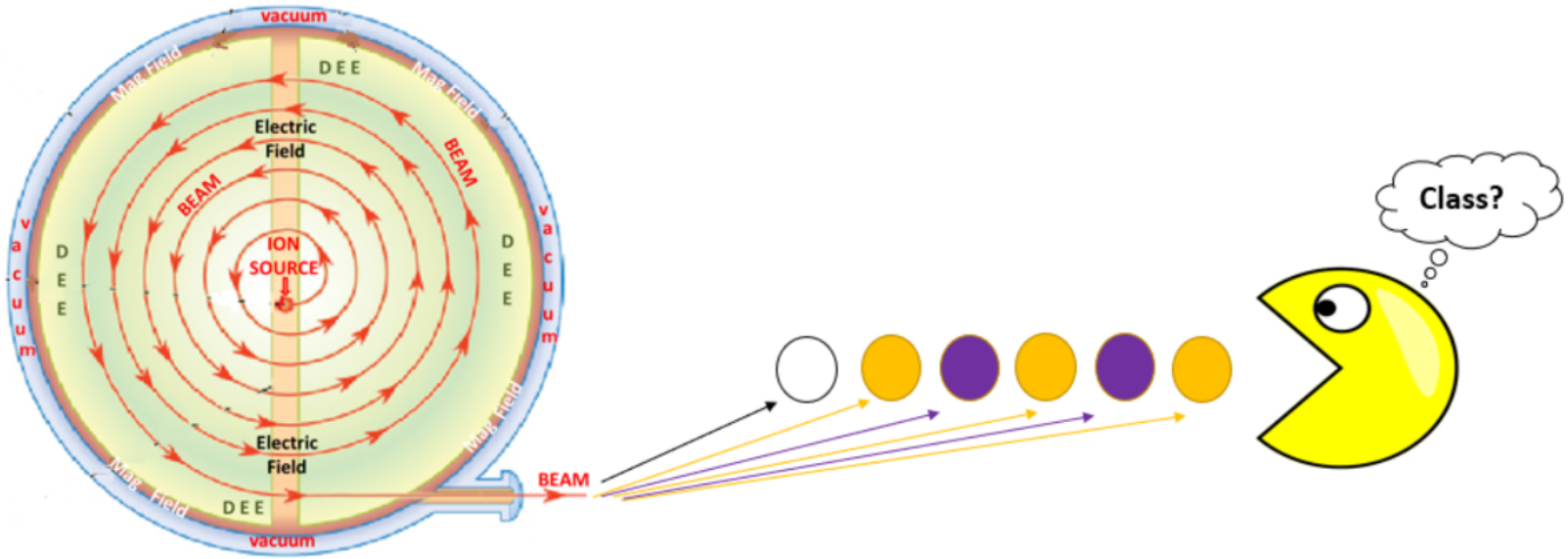
Joint Work with Xingye Qiao and Jiexin Duan

July 3, 2018

SUSY Data Set (UCI Machine Learning Repository)

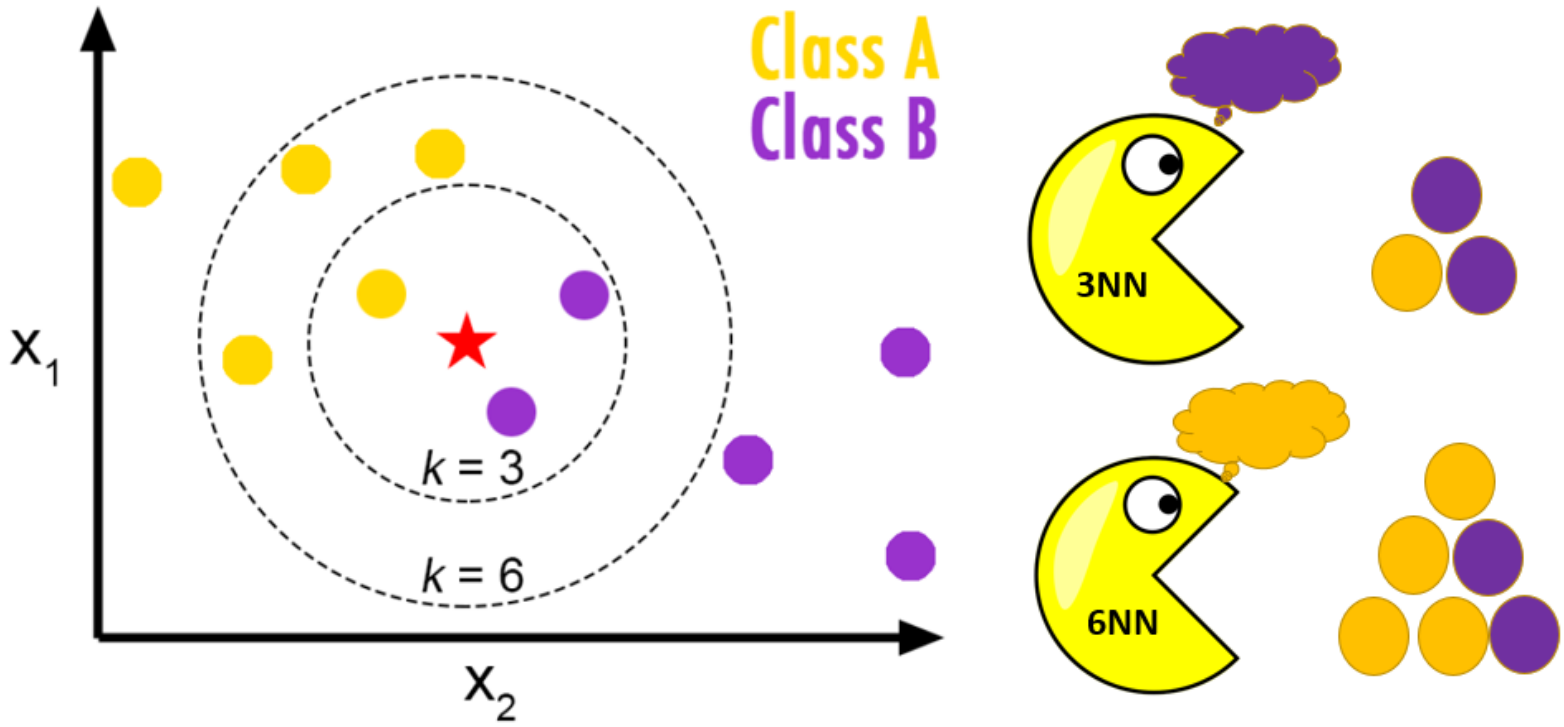
- Size: 5,000,000 particles from the accelerator
- Predictor variables: 18 (properties of the particles)
- Goal – Classification:

  distinguish between a signal process and a background process



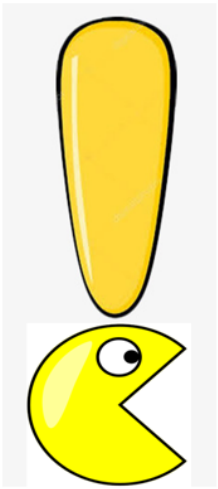
Nearest Neighbor Classification

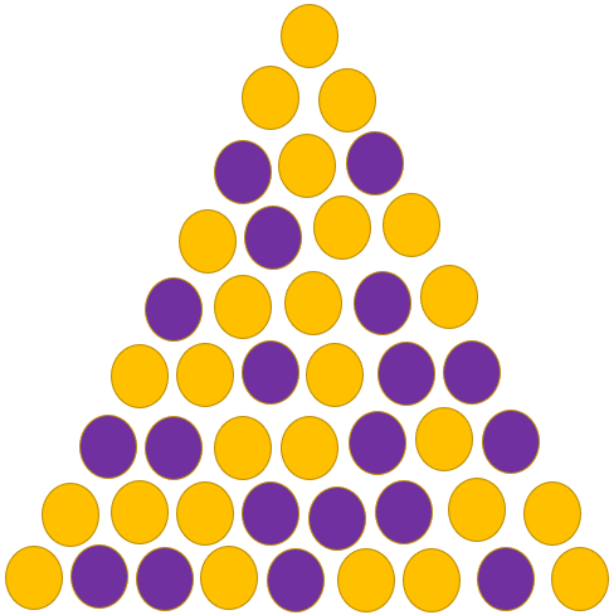
The kNN classifier predicts the class of $x \in \mathbb{R}^d$ to be the most frequent class of its k nearest neighbors (Euclidean distance).



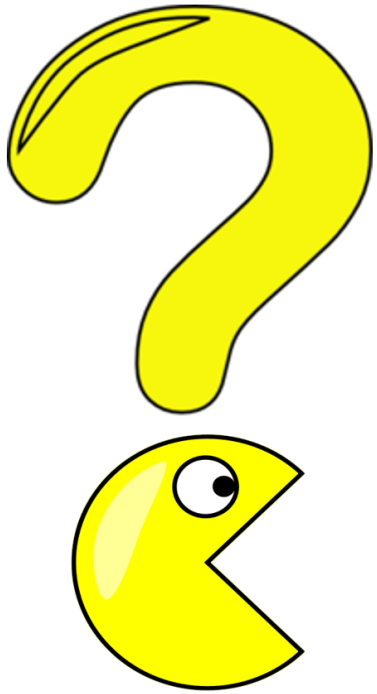
Computational/Space Challenges for kNN Classifiers

- Time complexity: $O(dN + Nlog(N))$
  - dN: computing distances from the query point to all N observations in $\mathbb{R}^d$
  - $Nlog(N)$: sorting N distances and selecting the k nearest 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 kNN

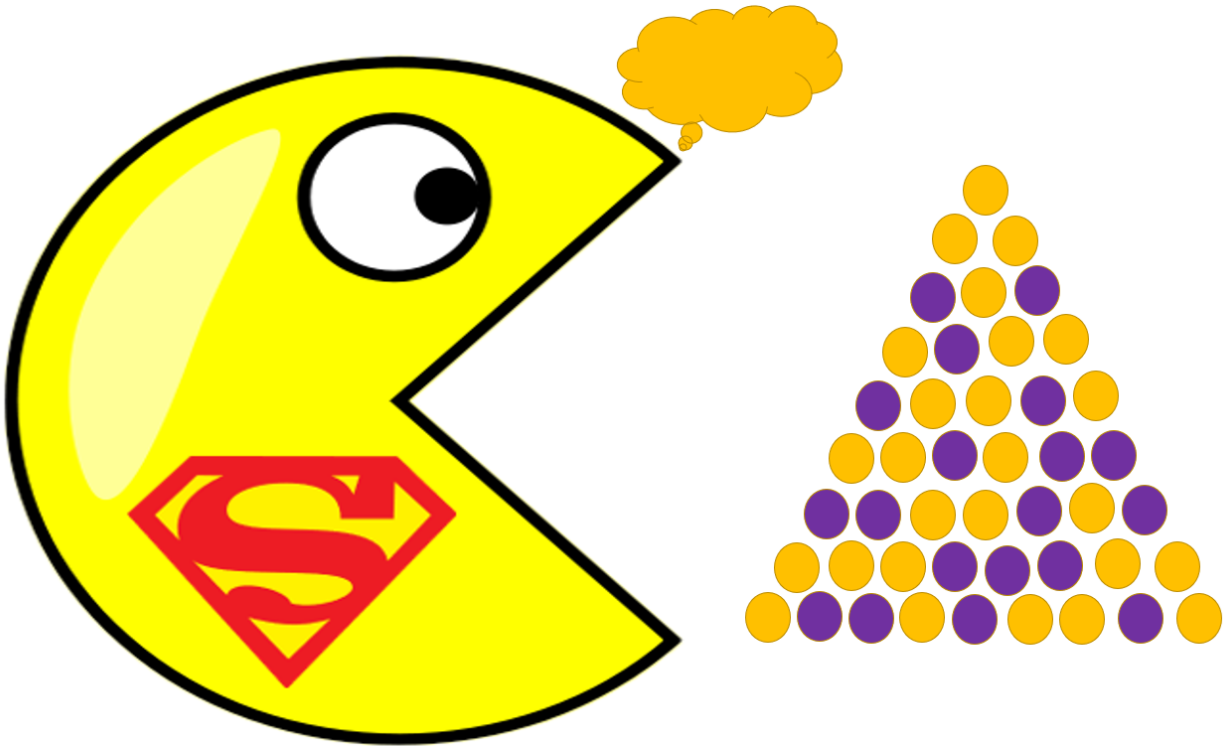
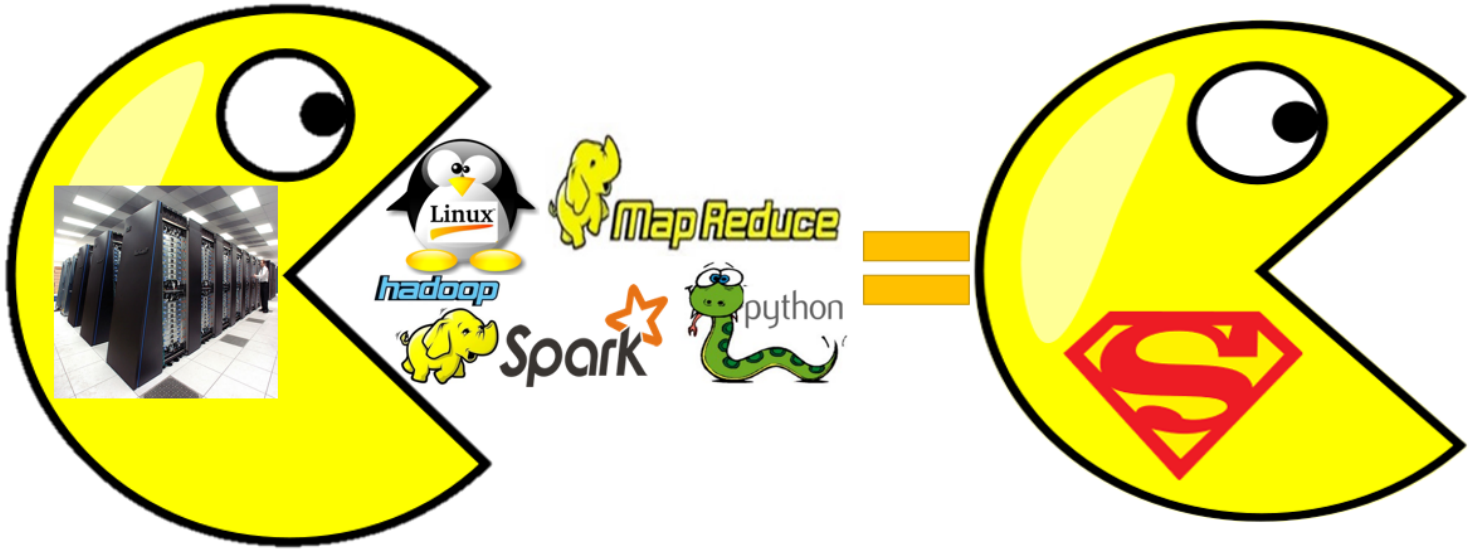


Figure: I'm Pac-Superman!

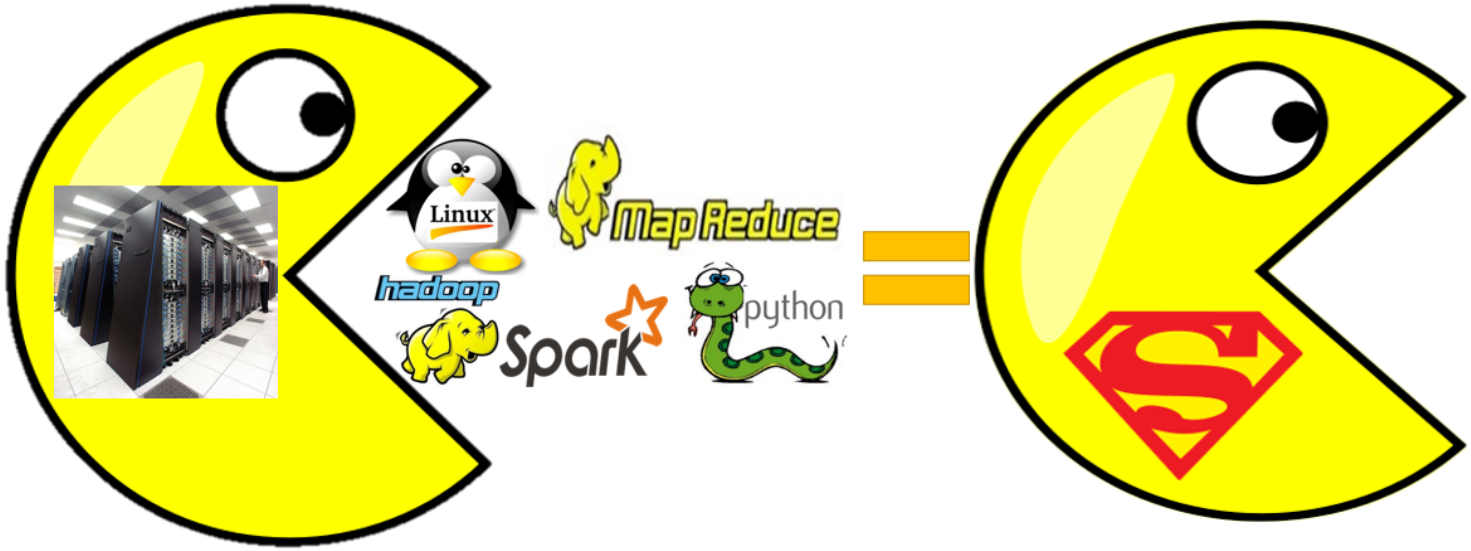
Construction of Pac-Superman



To build a Pac-superman for oracle-kNN, we need:

- Expensive super-computer

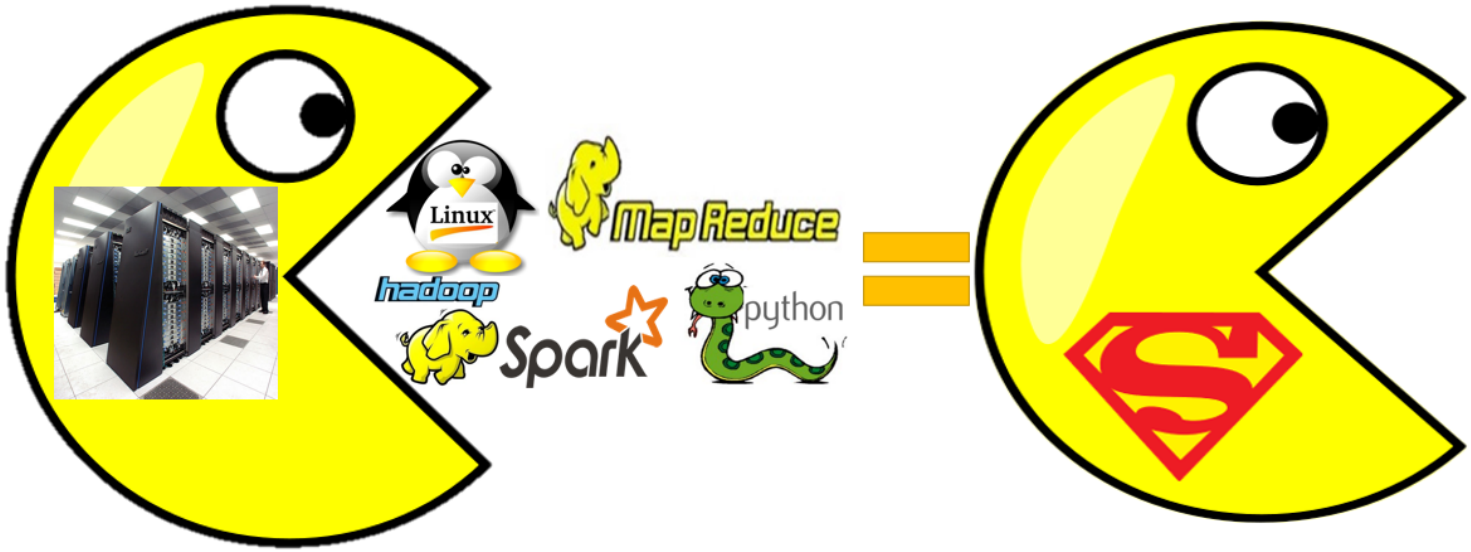
Construction of Pac-Superman



To build a Pac-superman for oracle-kNN, we need:

- Expensive super-computer
- Operation system, e.g. Linux

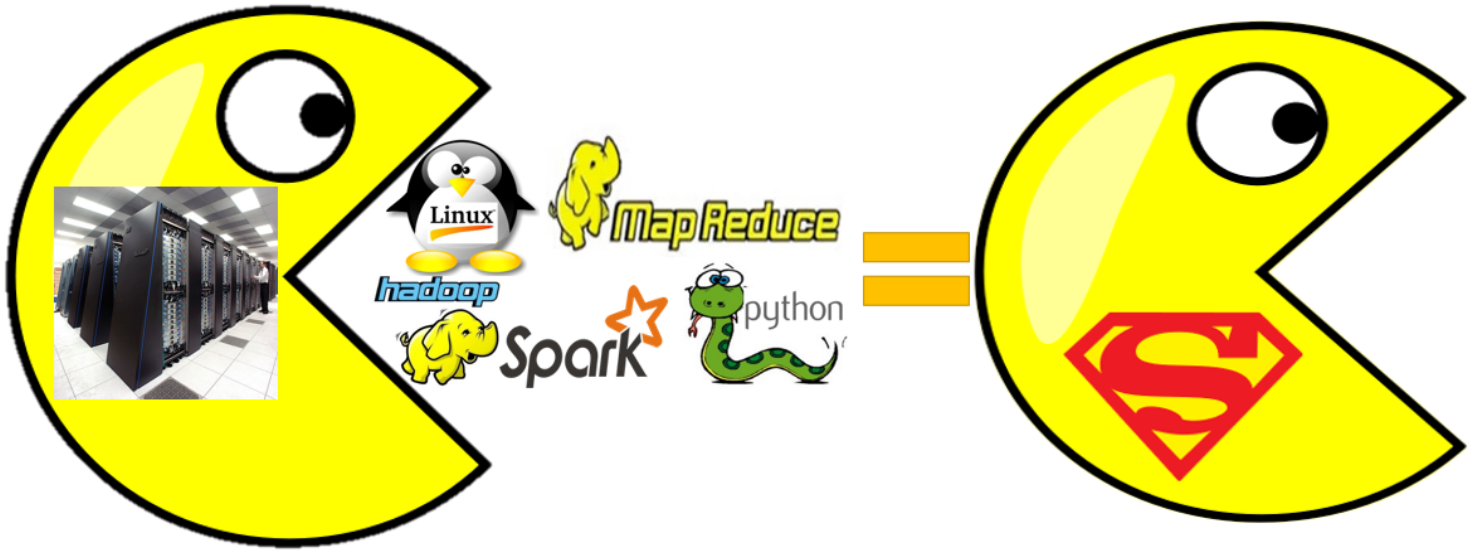
Construction of Pac-Superman



To build a Pac-superman for oracle-kNN, we need:

- Expensive super-computer
- Operation system, e.g. Linux
- Statistical software designed for big data, e.g. Spark, Hadoop

Construction of Pac-Superman



To build a Pac-superman for oracle-kNN, we need:

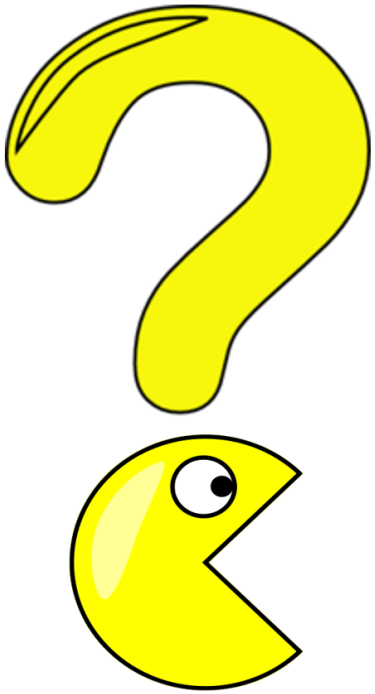
- Expensive super-computer
- Operation system, e.g. Linux
- Statistical software designed for big data, e.g. Spark, Hadoop
- Write complicated algorithms (e.g. MapReduce) with limited extendibility to other classification methods

Machine Learning Literature

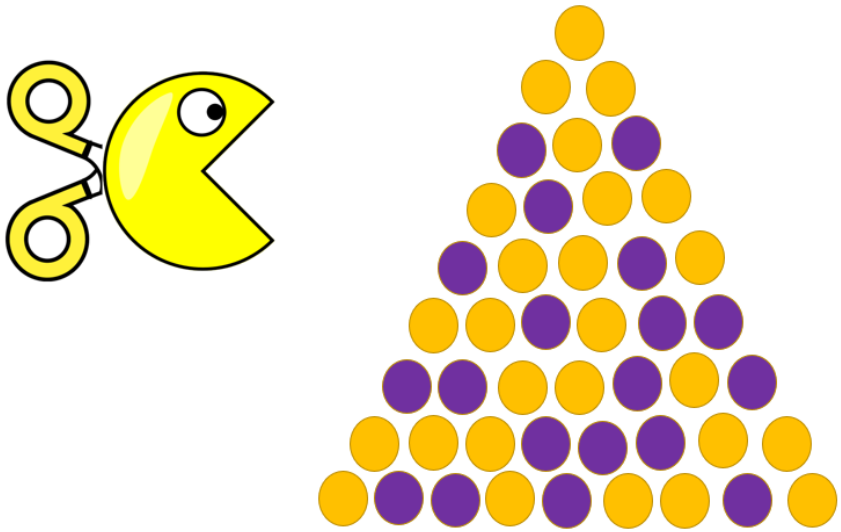
- In machine learning area, there are some nearest neighbor algorithms for big data already
  - Muja&Lowe(2014) designed an algorithm to search nearest neighbors for big data
  - Comments: complicated; without statistical gurantee; lack of extendibility
- We hope to design an algorithm framework:
  - Easy to implement
  - Strong statistical guarantee
  - More generic and expandable (e.g. easy to be applied to other classifiers (decision trees, SVM, etc)

If We Don't Have A Supercomputer...

What can we do?

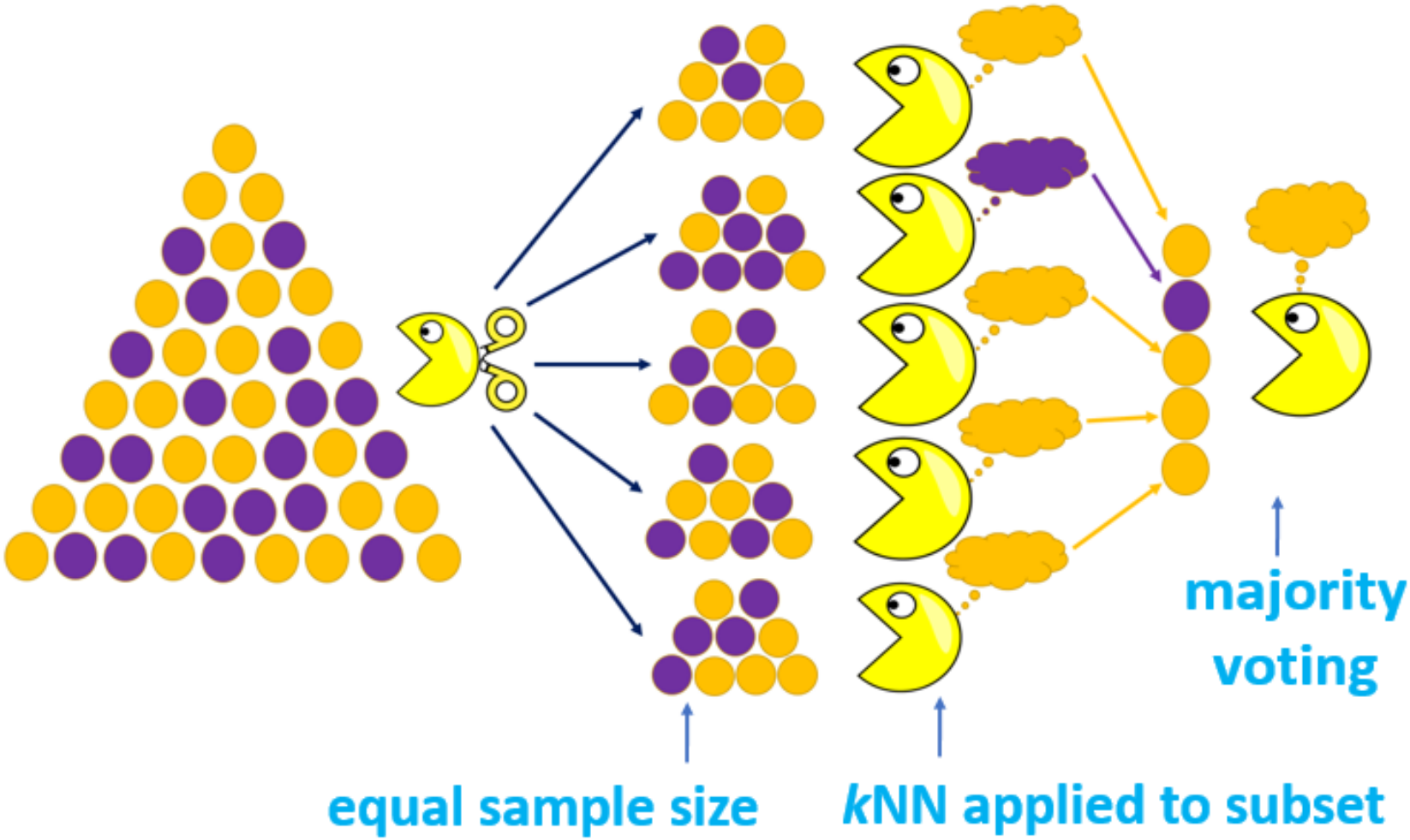


What about splitting the data?



Divide & Conquer (D&C) Framework

Goal: deliver *different* insights from regression problems.



Big-kNN for SUSY Data

- Oracle-kNN: kNN trained by the entire data
- Number of subsets: $s = N^{\gamma}$, $\gamma = 0.1, 0.2, \ldots 0.8$

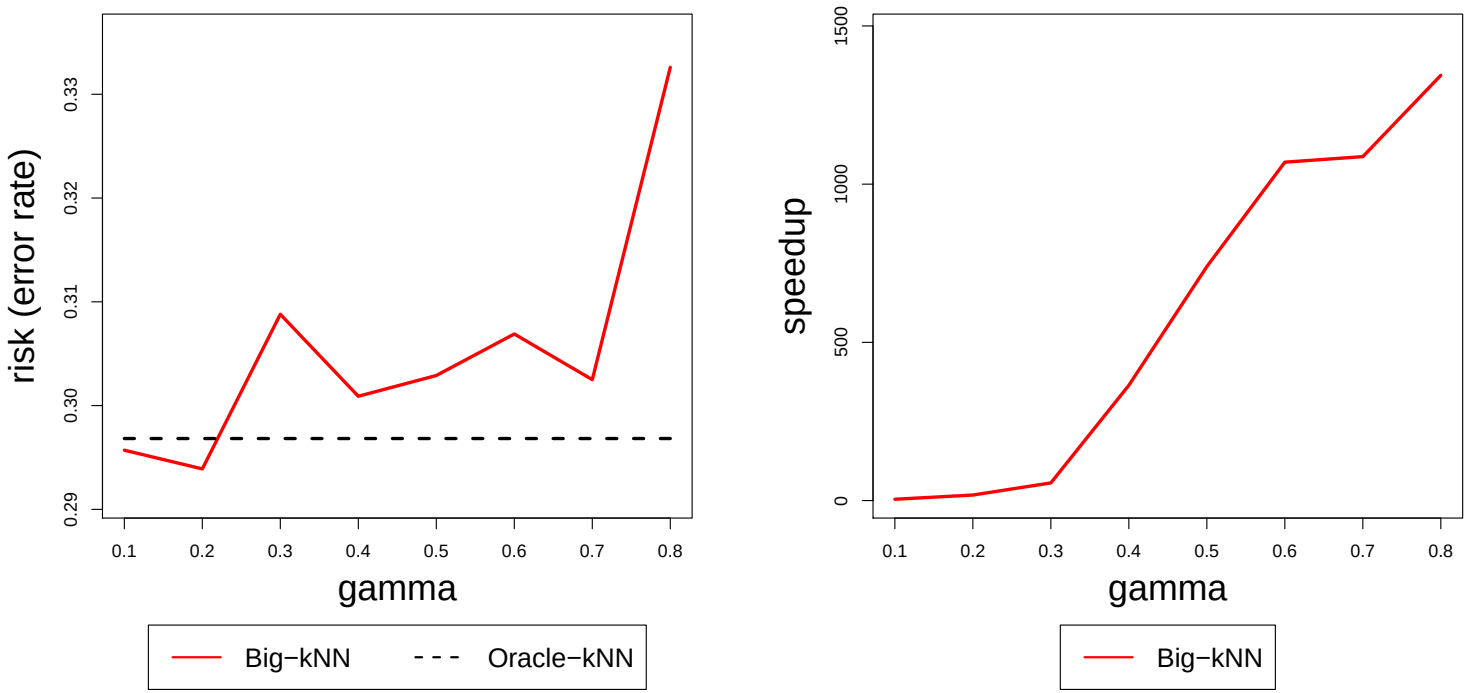
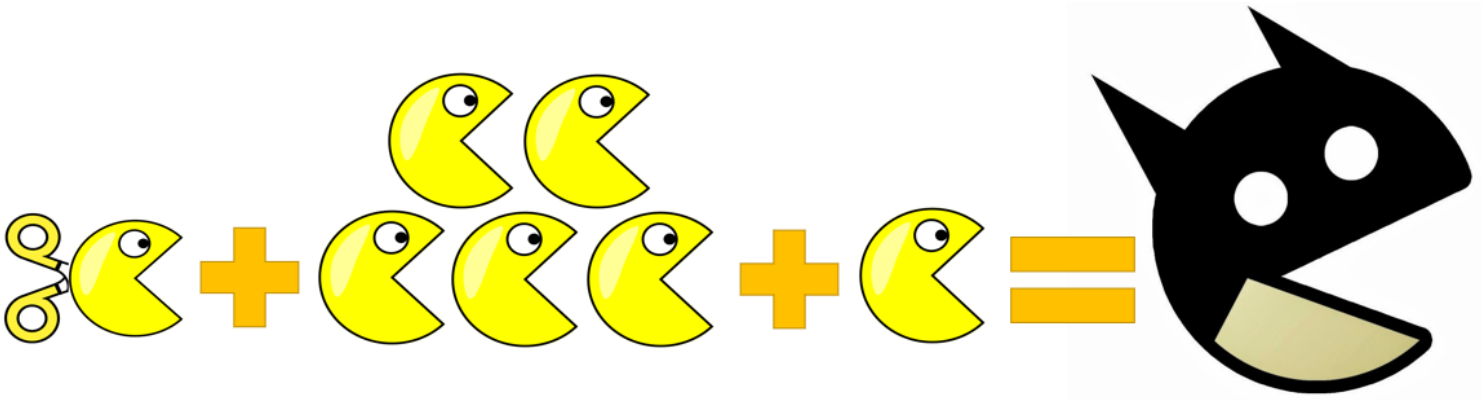


Figure: Risk and Speedup for Big-kNN.

- Risk of classifier ϕ: $R(\phi) = \mathbb{P}(\phi(X) \neq Y)$
- Speedup: running time ratio between Oracle-kNN & Big-kNN

Construction of Pac-Batman (Big Data)



We need a *statistical* understanding of Big-kNN....

Weighted Nearest Neighbor Classifiers (WNN)

Definition: WNN

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

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

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

Construction of Big-WNN

In a big data set with size $N = sn$, the Big-WNN is constructed as:

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

where $\widehat{\phi}_{n}^{(j)}(x)$ is the local WNN in j-th subset.

Statistical Guarantees: Classification Accuracy & Stability

- *Primary* criterion: accuracy
  - Regret=Expected Risk−Bayes Risk= $\mathbb{E}_{\mathcal{D}}\left[R(\widehat{\phi}_n)\right] - R(\phi^{\text{Bayes}})$
  - A small value of regret is preferred

Statistical Guarantees: Classification Accuracy & Stability

- *Primary* criterion: accuracy
  - Regret=Expected Risk−Bayes Risk= $\mathbb{E}_{\mathcal{D}}\left[R(\widehat{\phi}_n)\right] - R(\phi^{\text{Bayes}})$
  - A small value of regret is preferred
- *Secondary* criterion: stability (Sun et al, 2016, *JASA*)

Definition: Classification Instability (CIS)

Define classification instability of a classification procedure Ψ as

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

where $\widehat{\phi}_{n1}$ and $\widehat{\phi}_{n2}$ are the classifiers trained from the same classification procedure Ψ based on $\mathcal{D}_1$ and $\mathcal{D}_2$ with the same distribution.

- A small value of CIS is preferred

Asymptotic Regret of Big-WNN

Theorem (Regret of Big-WNN)

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

$$Regret\,(Big\text{-}WNN) \approx B_1 s^{-1} \sum_{i=1}^{n} w_{ni}^2 + B_2 \Big(\sum_{i=1}^{n} \frac{\alpha_i w_{ni}}{n^{2/d}} \Big)^2,$$

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

Asymptotic Regret of Big-WNN

Theorem (Regret of Big-WNN)

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

$$Regret\,(Big\text{-}WNN) \asymp B_1 s^{-1} \sum_{i=1}^{n} w_{ni}^2 + B_2 \Big(\sum_{i=1}^{n} \frac{\alpha_i w_{ni}}{n^{2/d}} \Big)^2,$$

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

Variance part

Asymptotic Regret of Big-WNN

Theorem (Regret of Big-WNN)

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

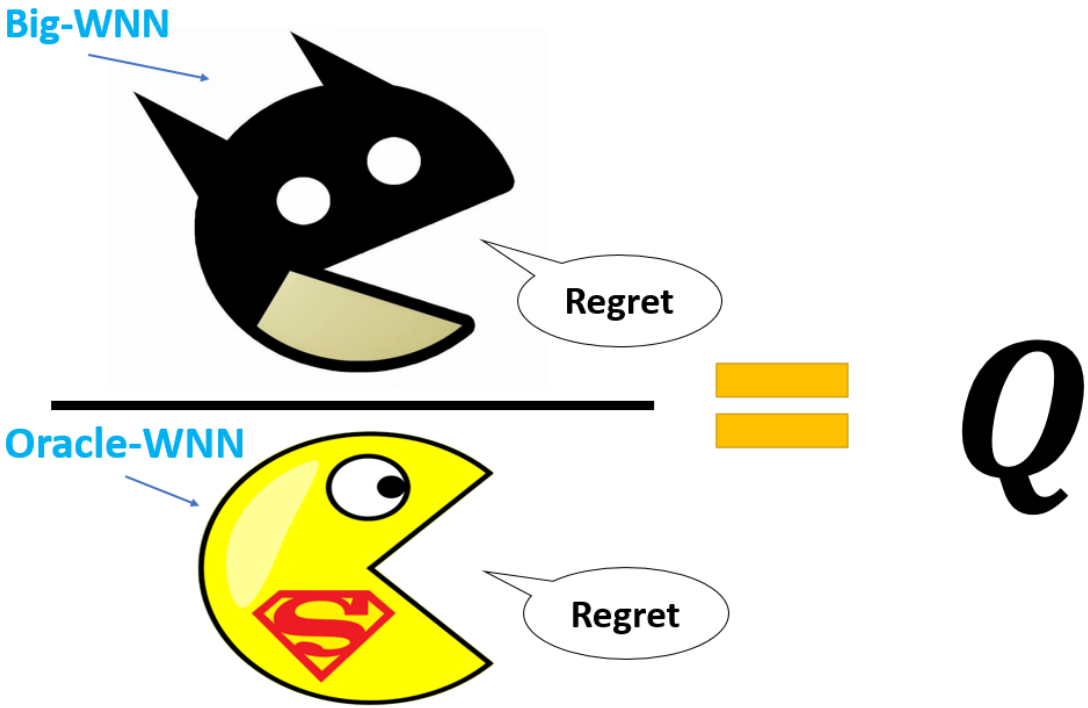
$$Regret\,(Big\text{-}WNN) \asymp B_1 s^{-1} \sum_{i=1}^{n} w_{ni}^2 + B_2 \Big(\sum_{i=1}^{n} \frac{\alpha_i w_{ni}}{n^{2/d}} \Big)^2,$$

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

Variance part

Bias part

Asymptotic Regret Comparison



Round 1

Asymptotic Regret Comparison

Theorem (Regret Ratio of Big-WNN and Oracle-WNN)

Given an Oracle-WNN classifier, we can always design a Big-WNN by adjusting its weights according to those in the oracle version s.t.

$$\frac{Regret\left(Big\text{-}WNN\right)}{Regret\left(Oracle\text{-}WNN\right)} \to Q,$$

where $Q = \left(\frac{\pi}{2}\right)^{\frac{4}{d+4}}$ *and* $1 < Q < 2$.

Asymptotic Regret Comparison

Theorem (Regret Ratio of Big-WNN and Oracle-WNN)

Given an Oracle-WNN classifier, we can always design a Big-WNN by adjusting its weights according to those in the oracle version s.t.

$$\frac{Regret\left(Big\text{-}WNN\right)}{Regret\left(Oracle\text{-}WNN\right)} \to Q,$$

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

- E.g., in Big-kNN case, we set $k = \lfloor (\frac{\pi}{2})^{\frac{d}{d+4}} \frac{k^O}{s} \rfloor$ given the oracle k^O. We need a multiplicative constant $(\frac{\pi}{2})^{\frac{d}{d+4}}$!

Asymptotic Regret Comparison

Theorem (Regret Ratio of Big-WNN and Oracle-WNN)

Given an Oracle-WNN classifier, we can always design a Big-WNN by adjusting its weights according to those in the oracle version s.t.

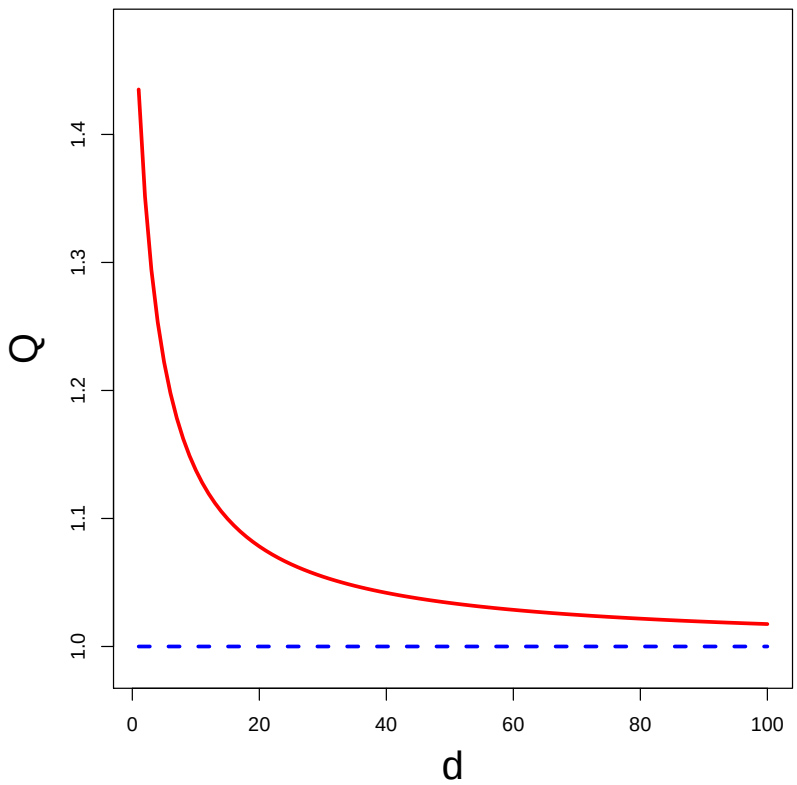
$$\frac{Regret\left(Big\text{-}WNN\right)}{Regret\left(Oracle\text{-}WNN\right)} \to Q,$$

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

- E.g., in Big-kNN case, we set $k = \lfloor (\frac{\pi}{2})^{\frac{d}{d+4}} \frac{k^O}{s} \rfloor$ given the oracle k^O. We need a multiplicative constant $(\frac{\pi}{2})^{\frac{d}{d+4}}$!
- We name Q the *Majority Voting (MV) Constant*.

Majority Voting Constant

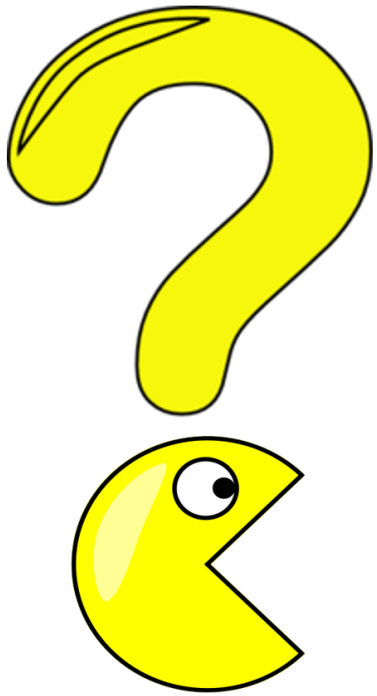
MV constant monotonically decreases to one as d increases



For example:

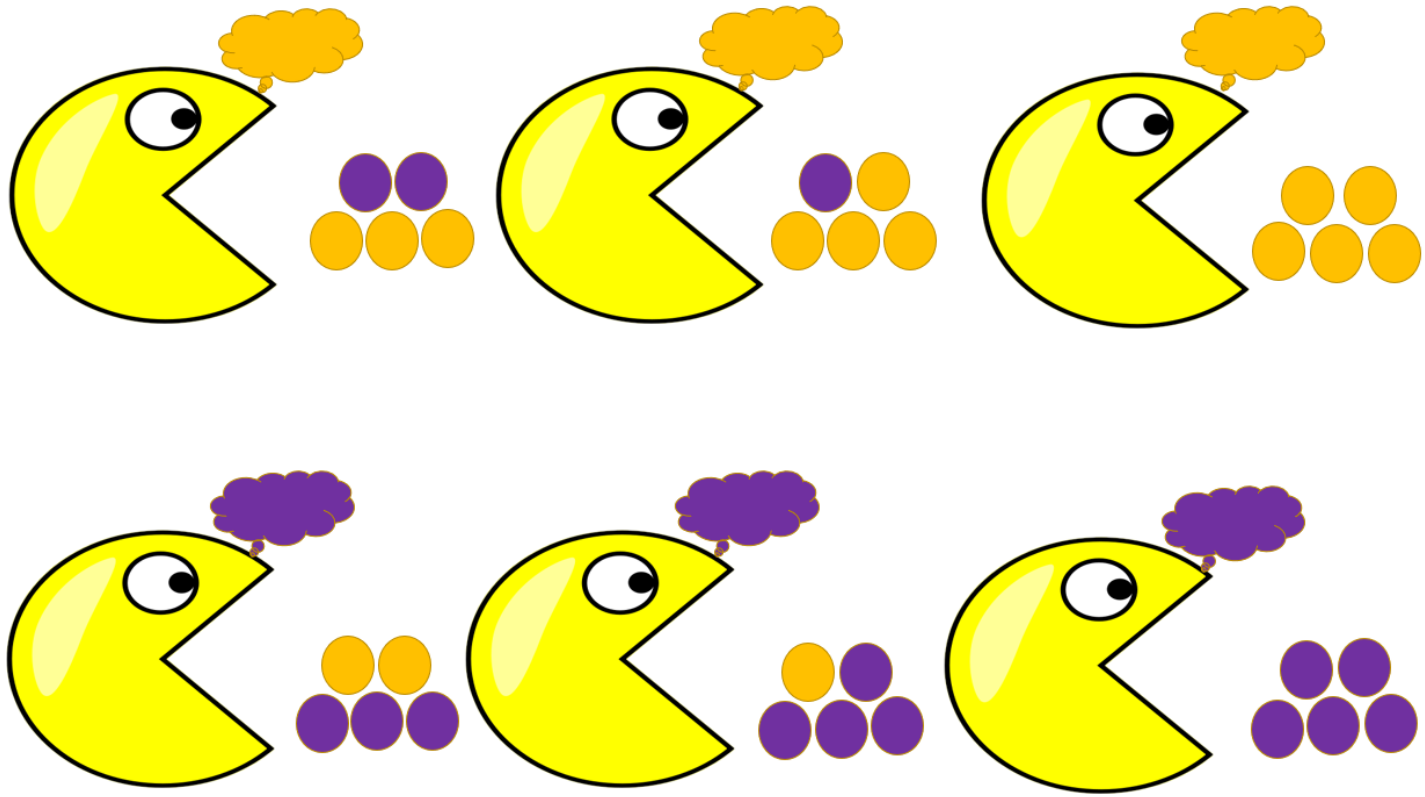
- d=1, Q=1.44
- d=2, Q=1.35
- d=5, Q=1.22
- d=10, Q=1.14
- d=20, Q=1.08
- d=50, Q=1.03
- d=100, Q=1.02

Why is there statistical accuracy loss?



Statistical Accuracy Loss due to Majority Voting

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



Only one majority voting in oracle classifier

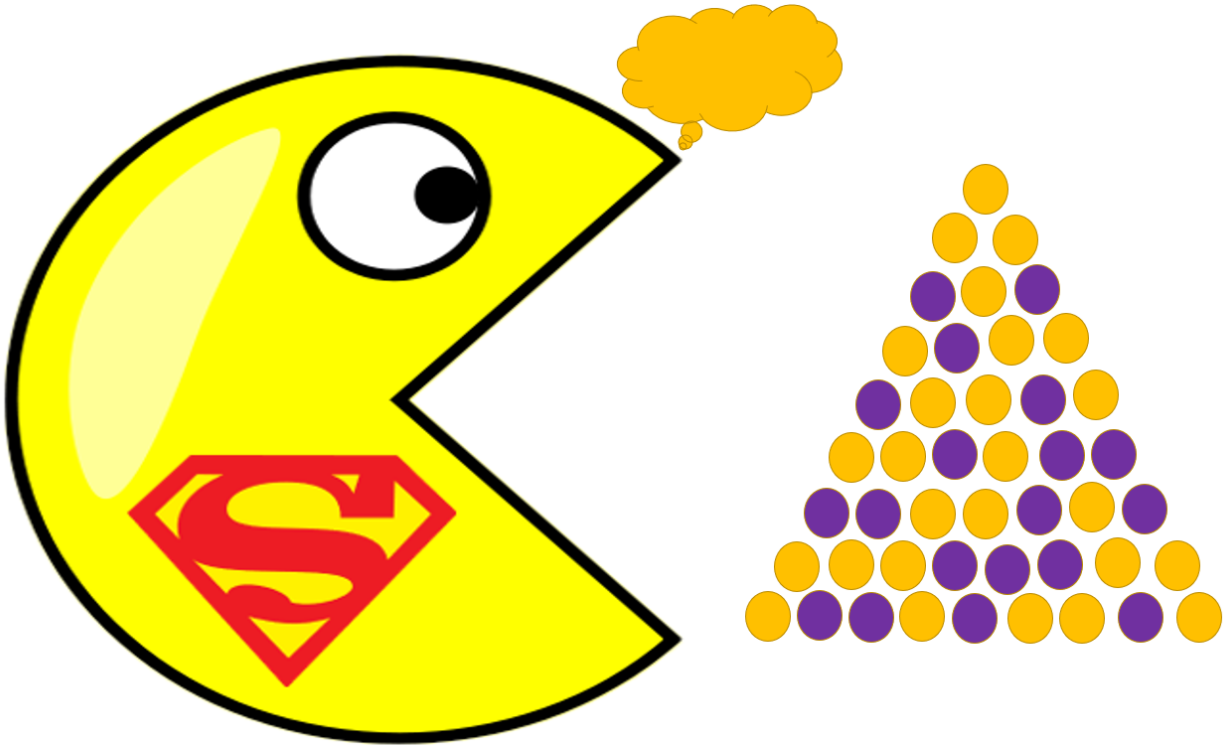
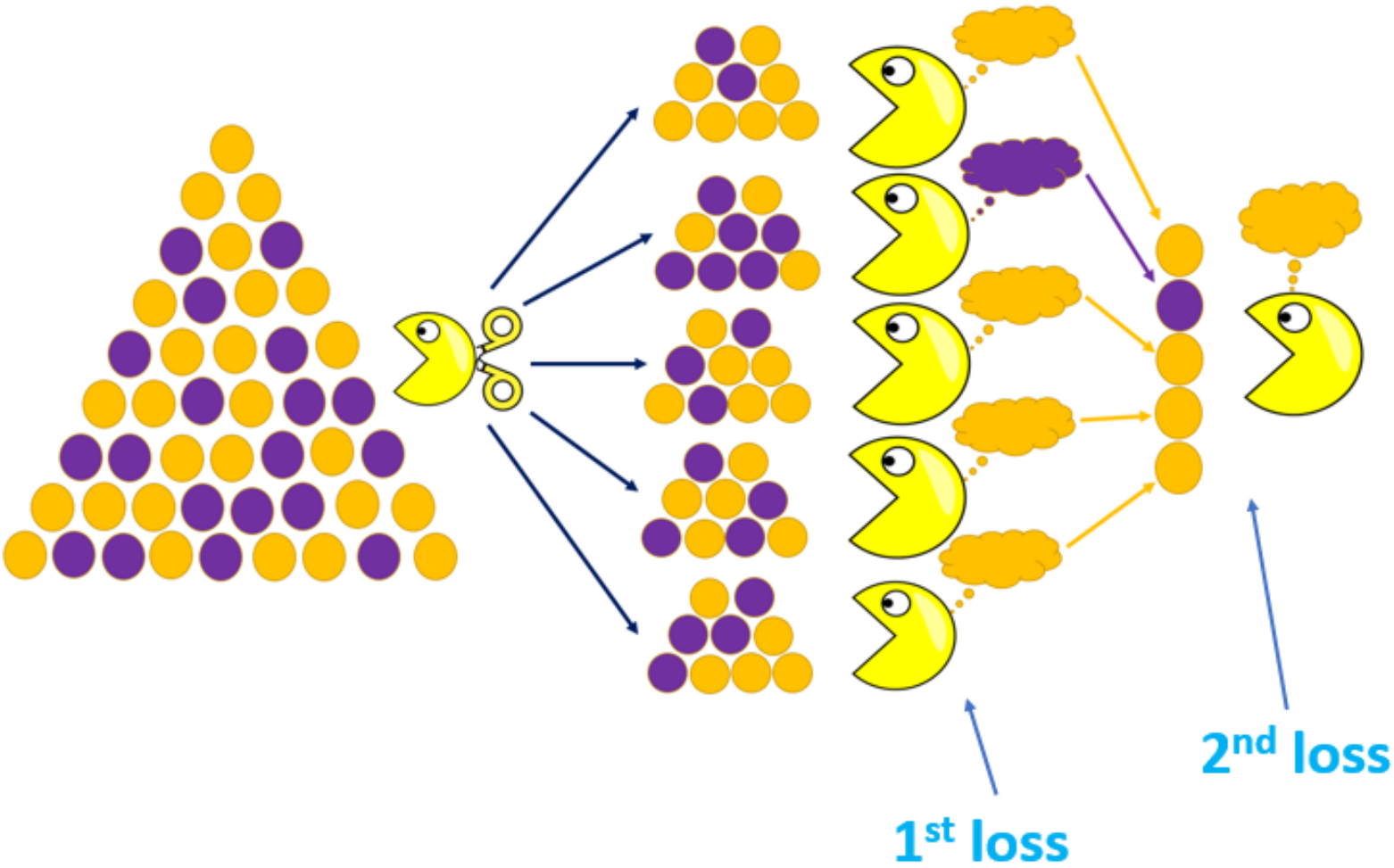
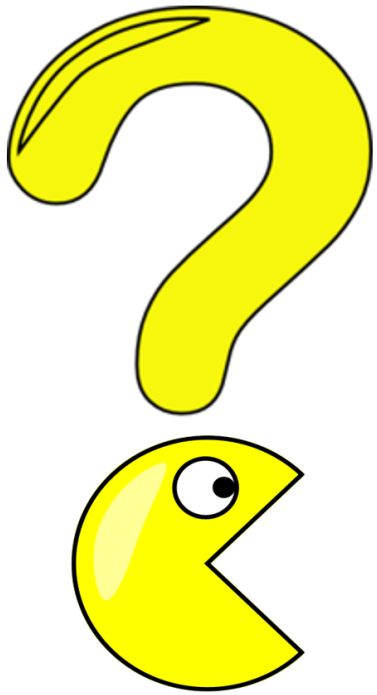


Figure: I'm Pac-Superman!

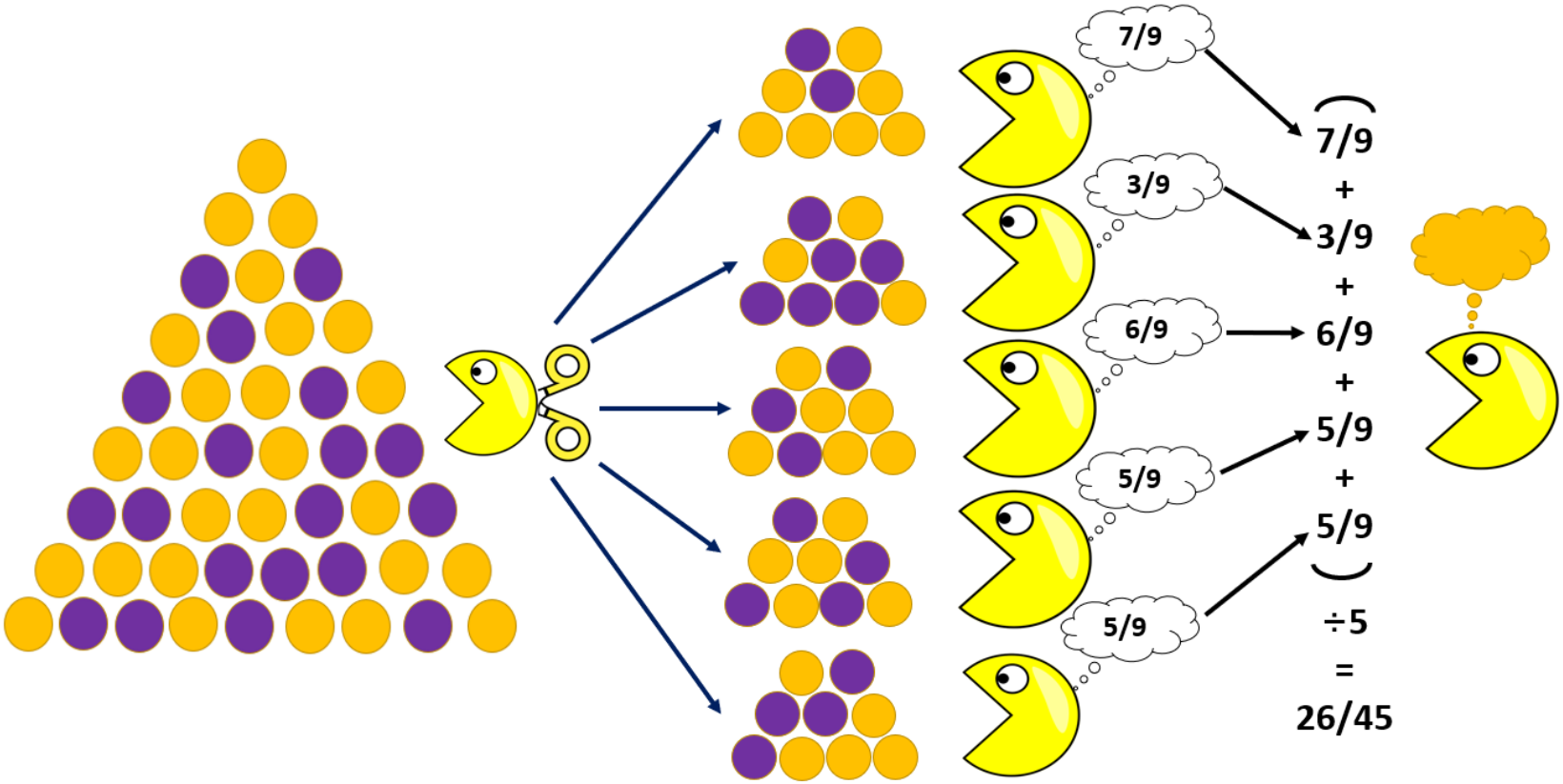
Majority Voting/Accuracy Loss Twice in D&C Framework



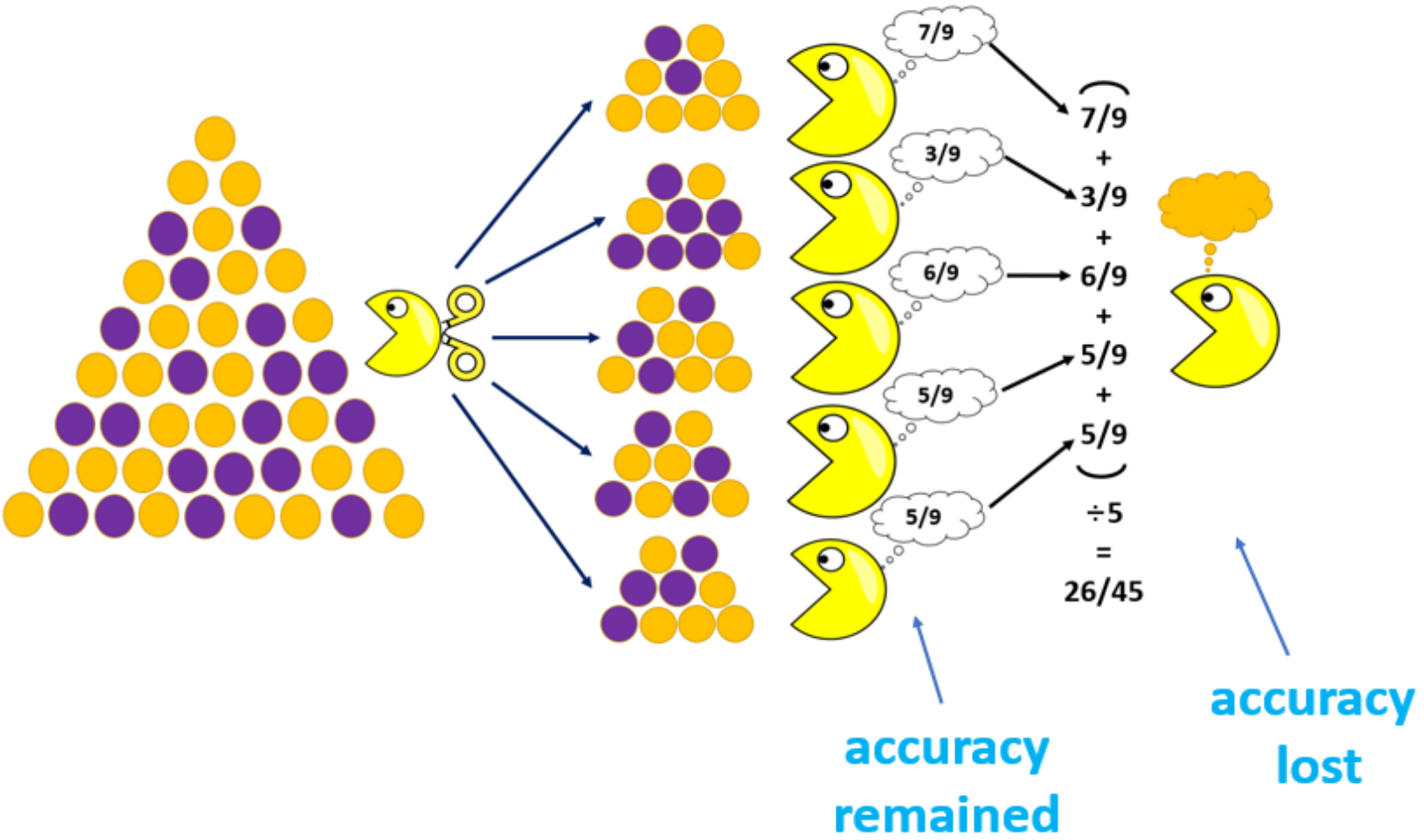
Is it possible to apply majority voting once in D&C?



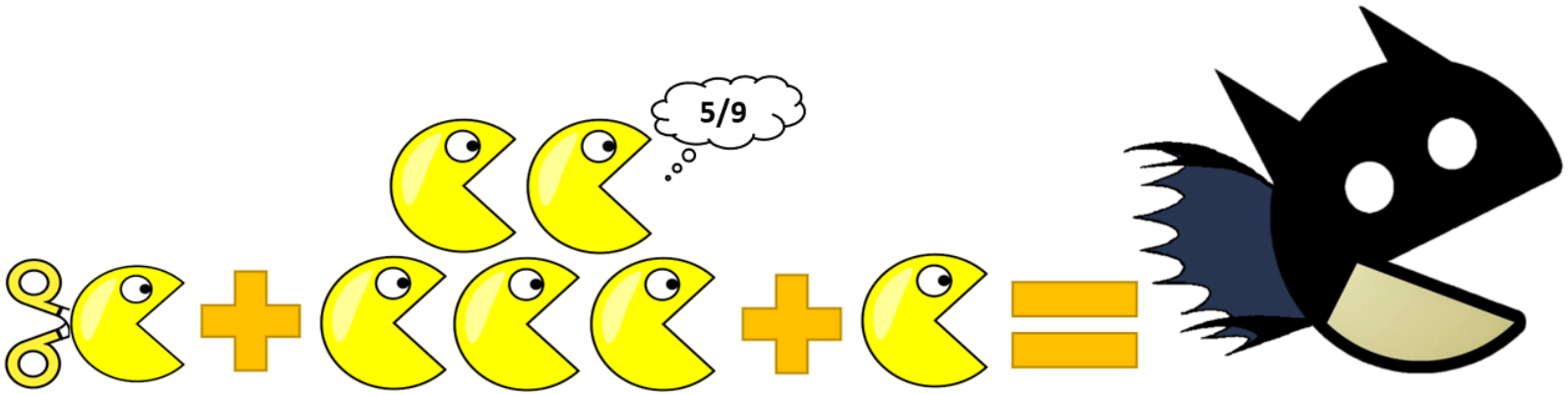
A Continuous Version of D&C Framework



Only Loss Once in the Continuous Version



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



Continuous Version of Big-WNN (C-Big-WNN)

Definition: C-Big-WNN

In a data set with size $N = sn$, the C-Big-WNN is constructed as:

$$\widehat{\phi}_{n,s}^{CBig}(x) = \mathbb{1}\left\{\frac{1}{s}\sum_{j=1}^{s}\widehat{S}_n^{(j)}(x) \geq 1/2\right\},$$

where $\widehat{S}_n^{(j)}(x) = \sum_{i=1}^{n} w_{ni,j}\, Y_{(i),j}(x)$ is the weighted average for nearest neighbors of query point x in j-th subset.

Continuous Version of Big-WNN (C-Big-WNN)

Definition: C-Big-WNN

In a data set with size $N = sn$, the C-Big-WNN is constructed as:

$$\widehat{\phi}_{n,s}^{CBig}(x) = \mathbb{1}\left\{\frac{1}{s}\sum_{j=1}^{s}\widehat{S}_n^{(j)}(x) \geq 1/2\right\},$$

where $\widehat{S}_n^{(j)}(x) = \sum_{i=1}^{n} w_{ni,j}\, Y_{(i),j}(x)$ is the weighted average for nearest neighbors of query point x in j-th subset.

Step 1:
Percentage calculation

Continuous Version of Big-WNN (C-Big-WNN)

Definition: C-Big-WNN

In a data set with size $N = sn$, the C-Big-WNN is constructed as:

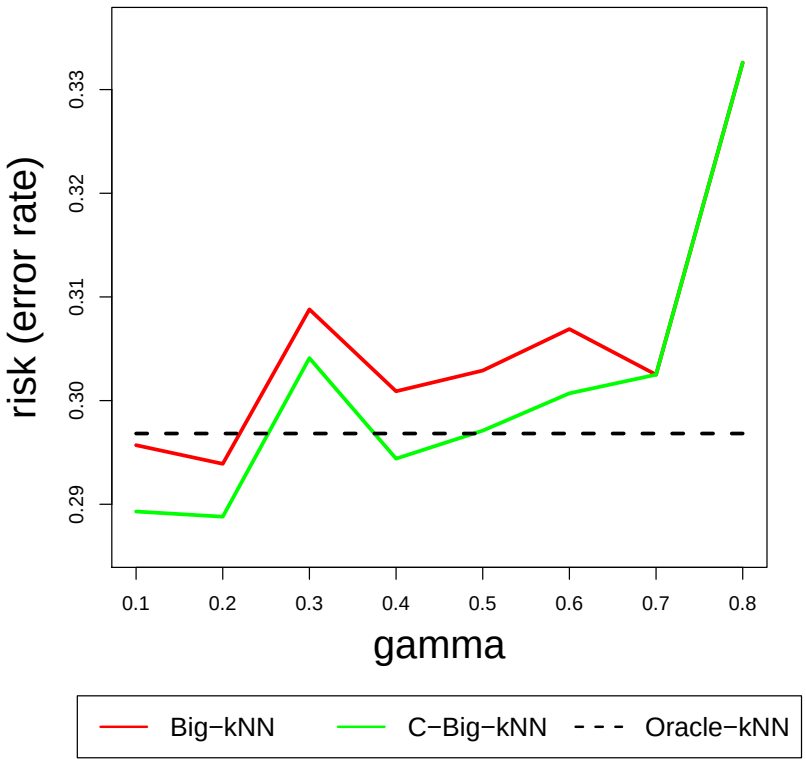
$$\widehat{\phi}_{n,s}^{CBig}(x) = \mathbb{1}\left\{\frac{1}{s}\sum_{j=1}^{s}\widehat{S}_n^{(j)}(x) \geq 1/2\right\},$$

where $\widehat{S}_n^{(j)}(x) = \sum_{i=1}^{n} w_{ni,j}\, Y_{(i),j}(x)$ is the weighted average for nearest neighbors of query point x in j-th subset.

Step 1:
Percentage calculation

Step 2:
Majority voting

C-Big-kNN for SUSY Data



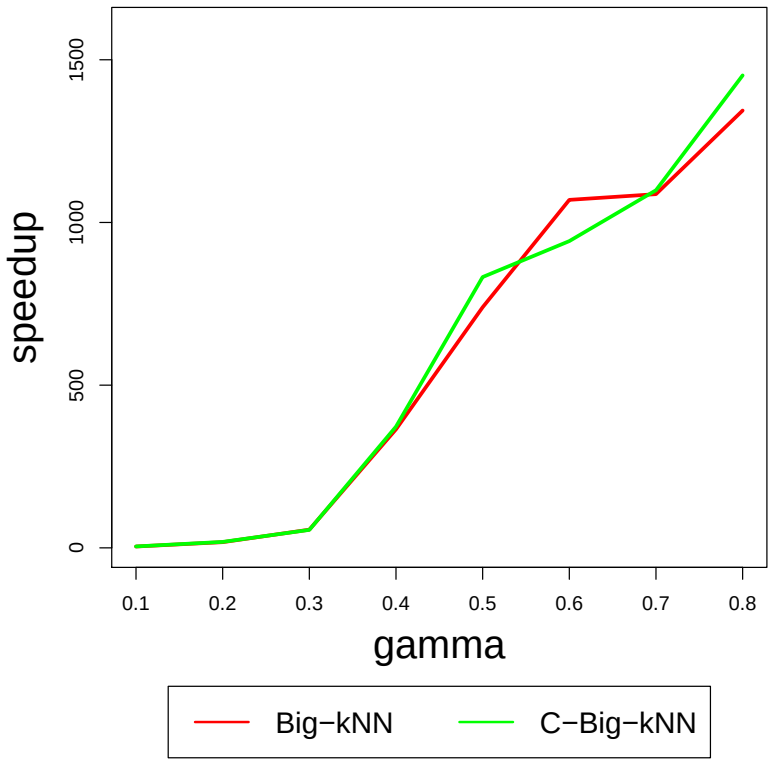
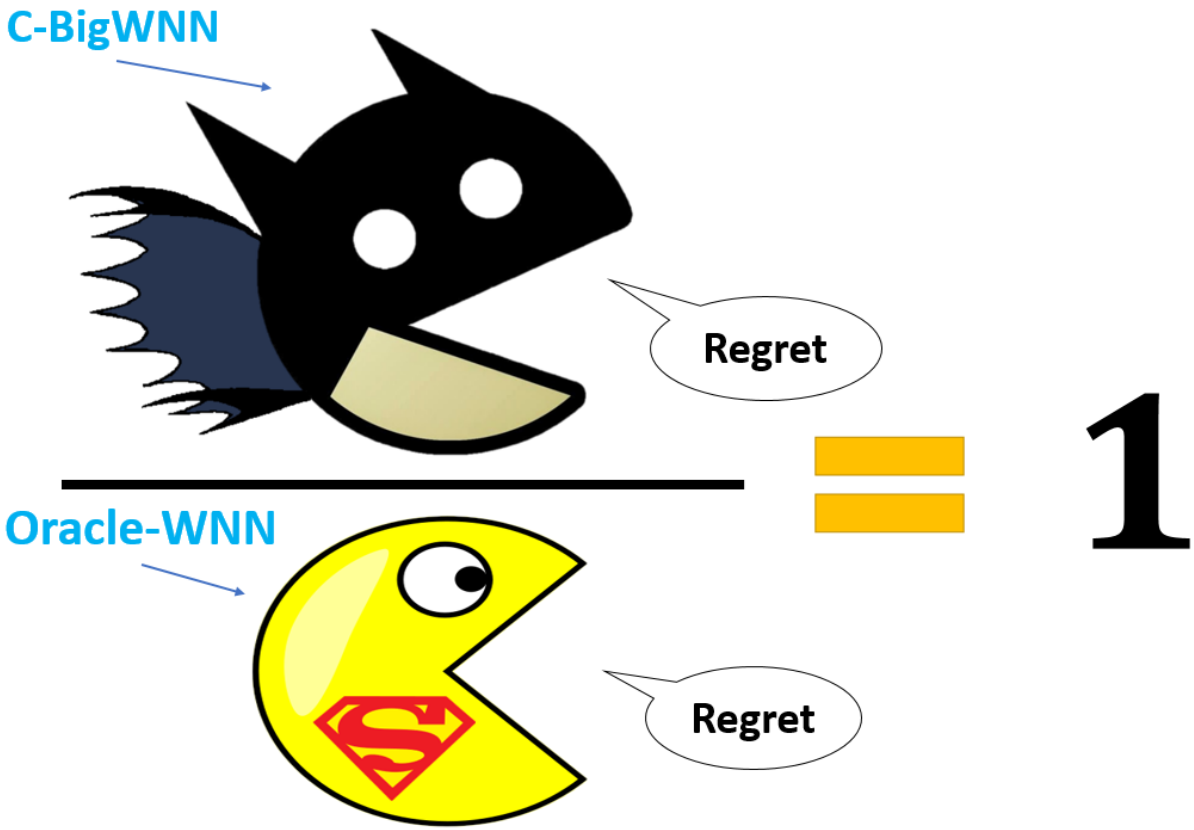


Figure: Risk and Speedup for Big-kNN and C-Big-kNN.

Asymptotic Regret Comparison



Round 2

Asymptotic Regret Comparison

Theorem (Regret Ratio between C-Big-WNN and Oracle-WNN)

Given an Oracle-WNN, we can always design a C-Big-WNN by adjusting its weights according to those in the oracle version s.t.

$$\frac{Regret\left(C\text{-}Big\text{-}WNN\right)}{Regret\left(Oracle\text{-}WNN\right)} \to 1,$$

Asymptotic Regret Comparison

Theorem (Regret Ratio between C-Big-WNN and Oracle-WNN)

Given an Oracle-WNN, we can always design a C-Big-WNN by adjusting its weights according to those in the oracle version s.t.

$$\frac{Regret\left(C\text{-}Big\text{-}WNN\right)}{Regret\left(Oracle\text{-}WNN\right)} \to 1,$$

- E.g., in the C-Big-kNN case, we can set $k = \lfloor \frac{k^O}{s} \rfloor$ given the oracle k^O. There is no multiplicative constant!

Asymptotic Regret Comparison

Theorem (Regret Ratio between C-Big-WNN and Oracle-WNN)

Given an Oracle-WNN, we can always design a C-Big-WNN by adjusting its weights according to those in the oracle version s.t.

$$\frac{Regret\left(C\text{-}Big\text{-}WNN\right)}{Regret\left(Oracle\text{-}WNN\right)} \to 1,$$

- E.g., in the C-Big-kNN case, we can set $k = \lfloor \frac{k^O}{s} \rfloor$ given the oracle k^O. There is no multiplicative constant!
- Note that this theorem can be directly applied to the optimally weighted NN (Samworth, 2012, *AoS*), i.e., OWNN, whose weights are optimized to lead to the minimal regret.

Practical Consideration: Speed vs Accuracy

- It is time consuming to apply OWNN in each subset although it leads to the minimal regret

Practical Consideration: Speed vs Accuracy

- It is time consuming to apply OWNN in each subset although it leads to the minimal regret
- Rather, it is more easier and faster to implement kNN in each subset (with accuracy loss, though)

Practical Consideration: Speed vs Accuracy

- It is time consuming to apply OWNN in each subset although it leads to the minimal regret
- Rather, it is more easier and faster to implement kNN in each subset (with accuracy loss, though)
- What is the statistical price (in terms of accuracy and stability) in order to trade accuracy with speed?

Theorem (Statistical Price)

We can design an optimal C-Big-kNN by setting its weights as $k = \lfloor \frac{k^{O,opt}}{s} \rfloor$ *s.t.*

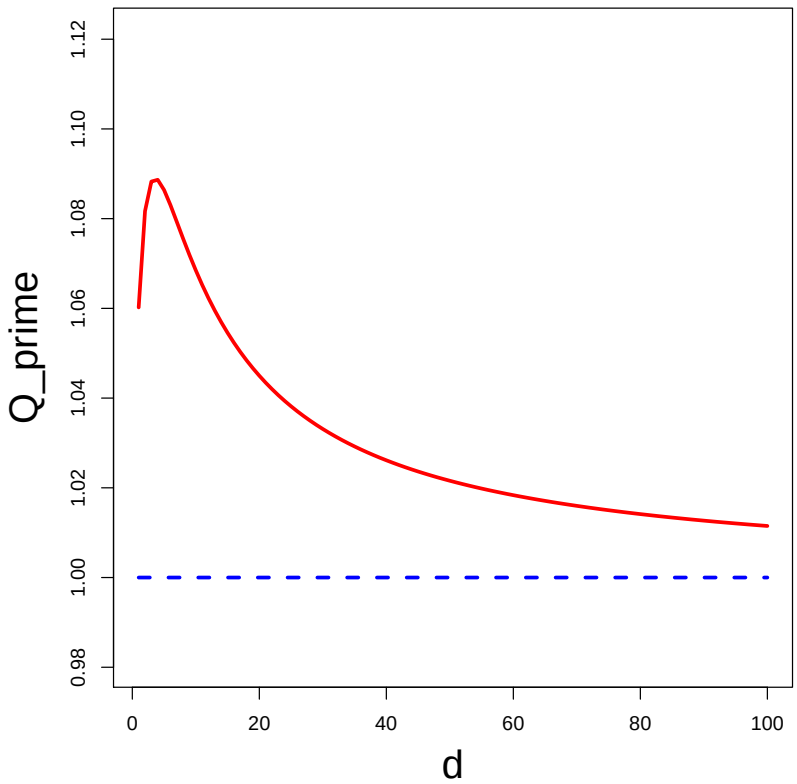
$$\frac{Regret(\,Optimal\ C\text{-}Big\text{-}kNN)}{Regret(\,Oracle\text{-}OWNN)} \to Q',$$

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

where $Q' = 2^{-4/d+4}(\frac{d+4}{d+2})^{(2d+4)/(d+4)}$ *and* $1 < Q' < 2$. *Here,* $k^{O,opt}$ *minimizes the regret of Oracle-kNN.*

Interestingly, both Q and Q' only depend on data dimension!

Q' converges to one as d grows in a unimodal way



For example:

- d=1, Q=1.06
- d=2, Q=1.08
- d=5, Q=1.08
- d=10, Q=1.07
- d=20, Q=1.04
- d=50, Q=1.02
- d=100, Q=1.01

The worst dimension is $d^* = 4 \Longrightarrow Q^* = 1.089$.

Simulation Analysis – kNN

Consider the classification problem for Big-kNN and C-Big-kNN:

- Sample size: $N = 27,000$
- Dimensions: $d = 6, 8$
- $P_0 \sim N(0_d, \mathbb{I}_d)$ 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-kNN: $k^O = N^{0.7}$
- Number of subsamples in D&C: $s = N^\gamma$, $\gamma = 0.1, 0.2, \ldots 0.8$
- Number of neighbors in Big-kNN: $k^d = \lfloor (\frac{\pi}{2})^{\frac{d}{d+4}} \frac{k^O}{s} \rfloor$
- Number of neighbors in C-Big-kNN: $k^c = \lfloor \frac{k^O}{s} \rfloor$

Simulation Analysis – Empirical Risk

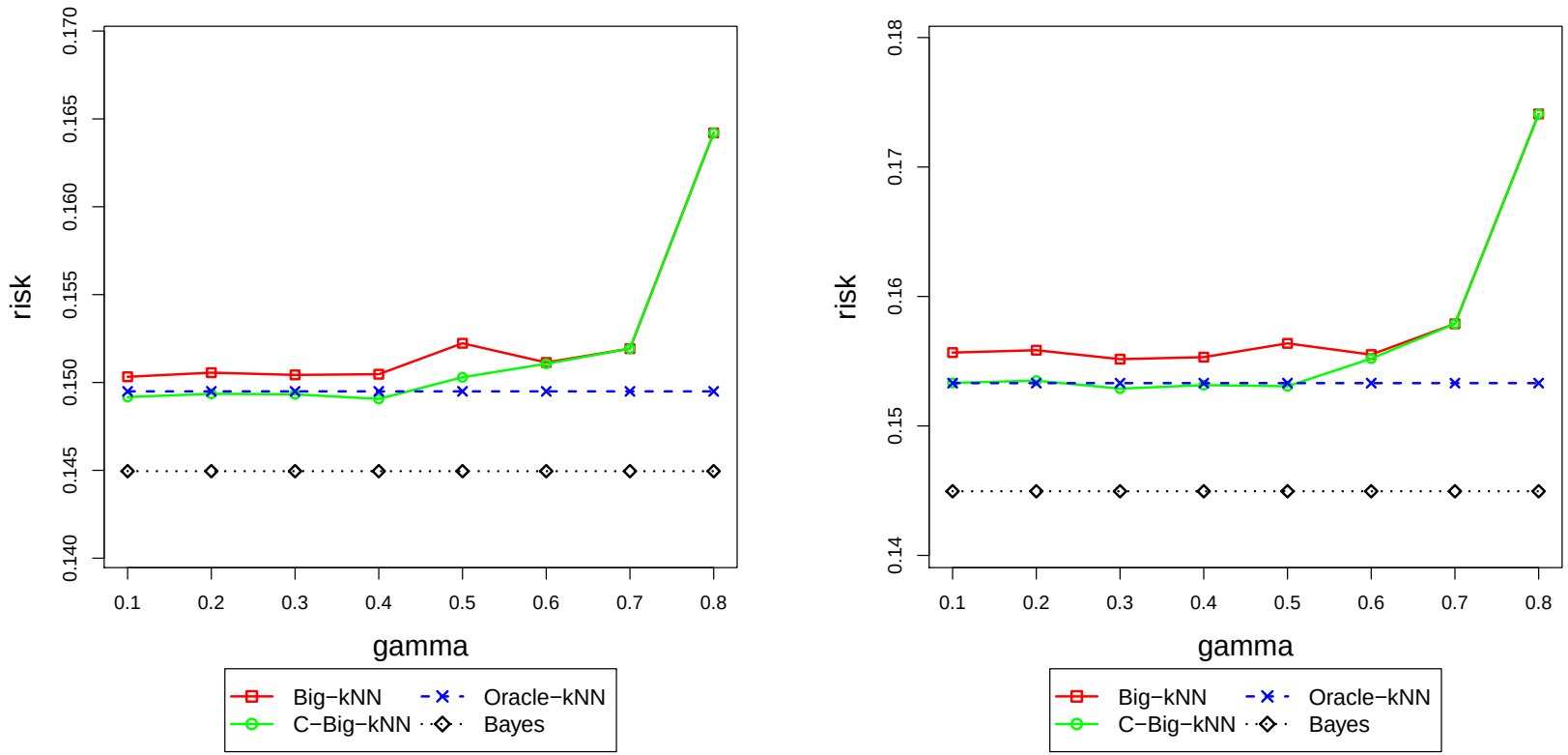


Figure: Empirical Risk (Testing Error). Left/Right: $d = 6/8$.

Simulation Analysis – Running Time

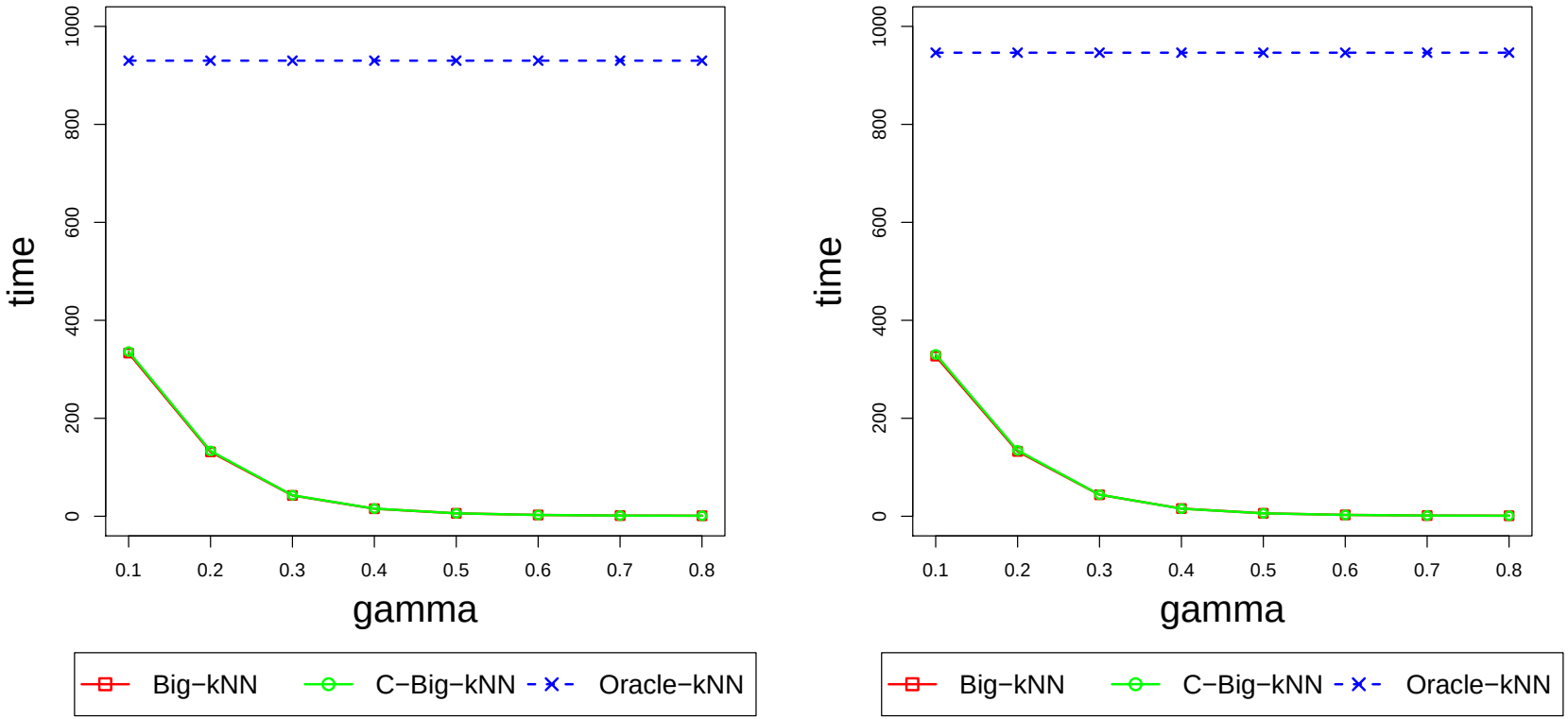


Figure: Running Time. Left/Right: $d = 6/8$.

Simulation Analysis – Empirical Regret Ratio

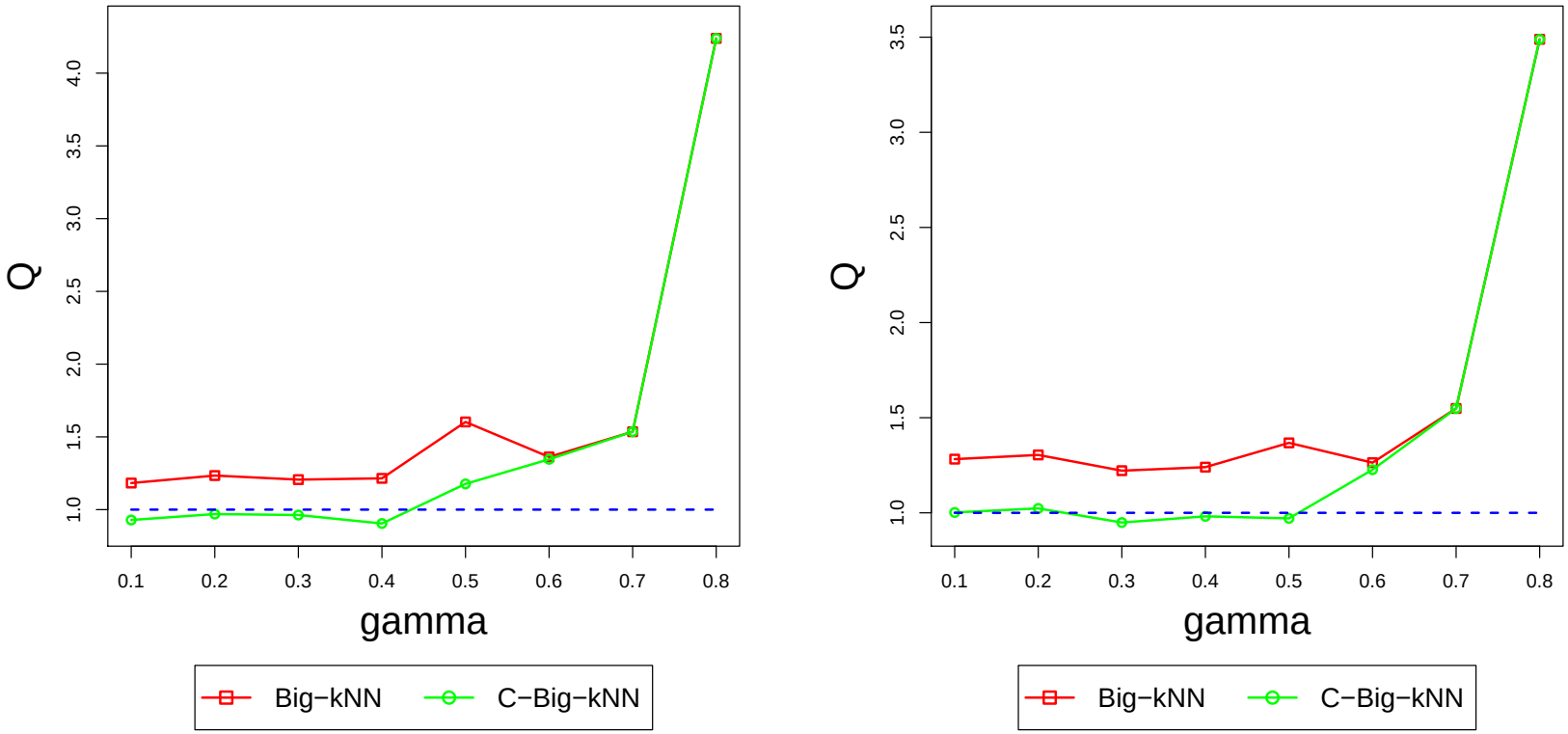


Figure: Empirical Ratio of Regret. Left/Right: $d = 6/8$.

- Q: Regret(Big-kNN)/Regret(Oracle-kNN) or Regret(C-Big-kNN)/Regret(Oracle-kNN)

Simulation Analysis – Empirical CIS

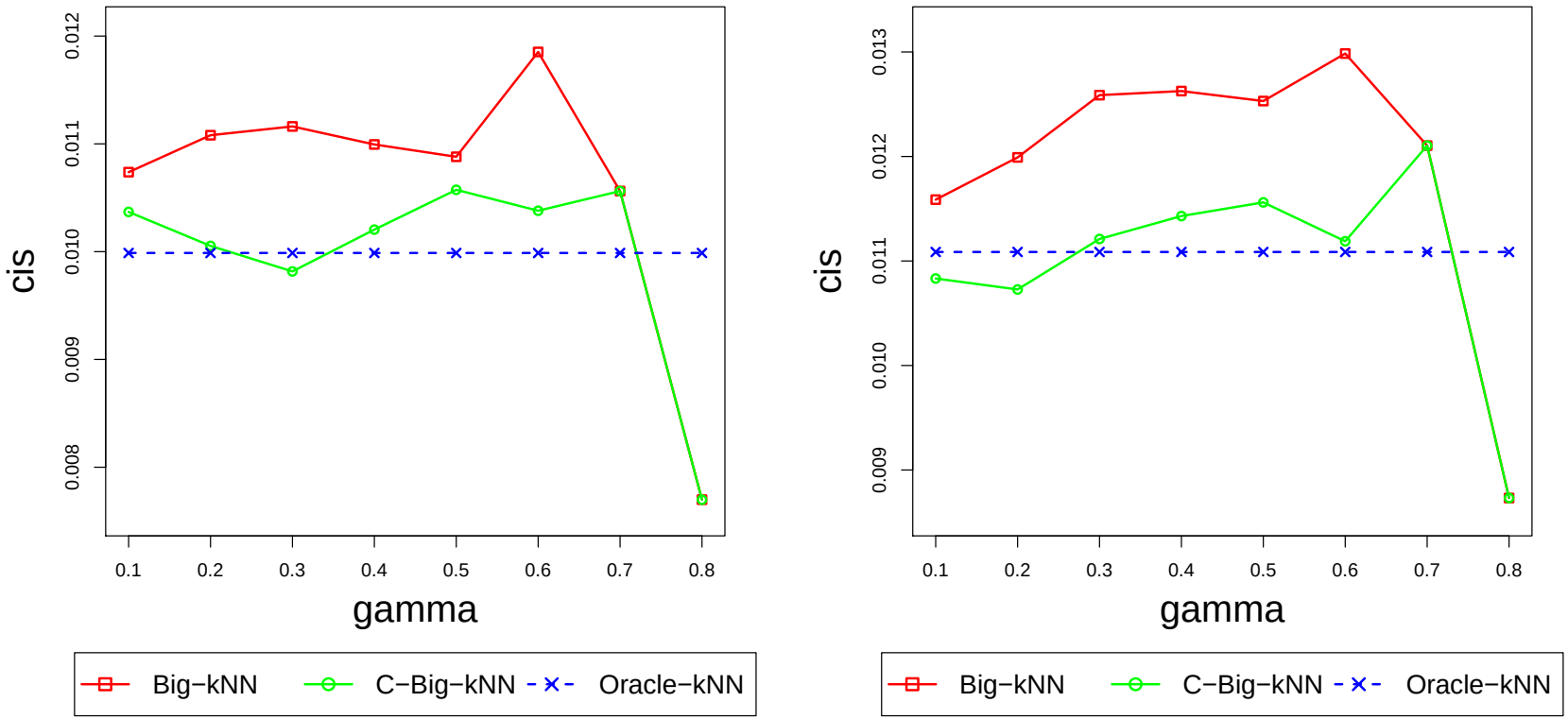
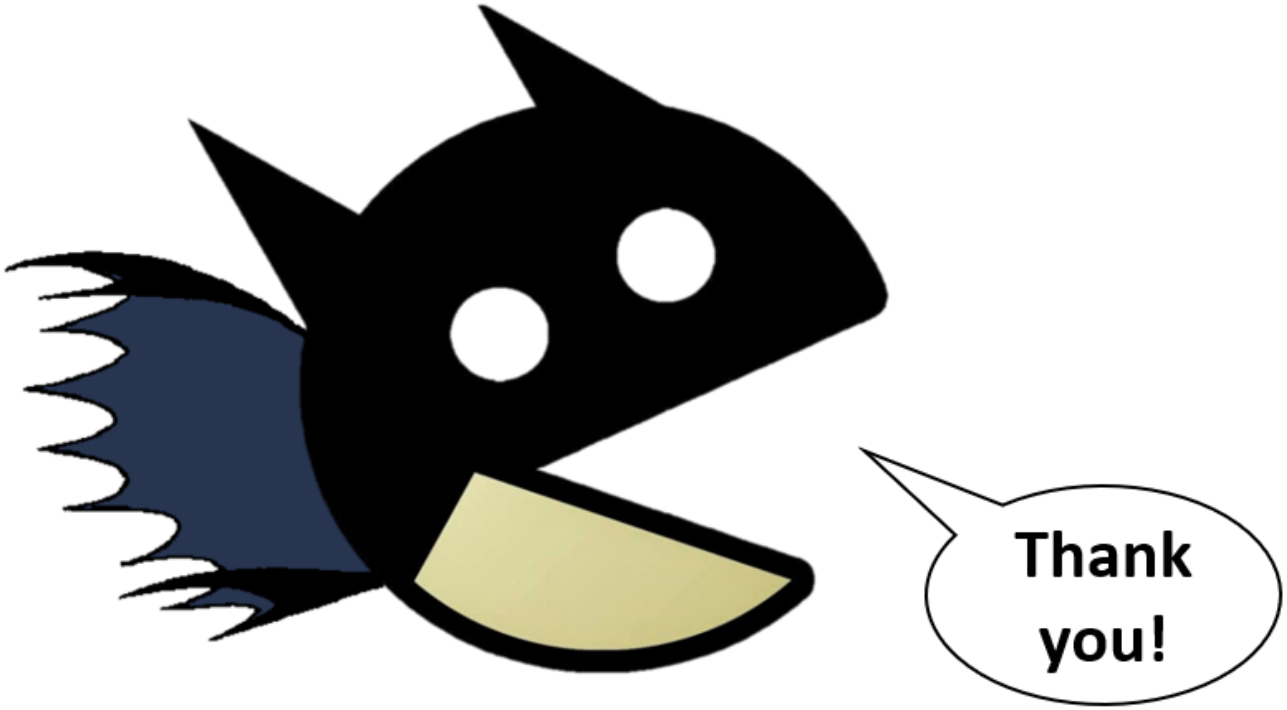


Figure: Empirical CIS. Left/Right: $d = 6/8$.

Real Data Analysis – UCI Machine Learning Repository

Data	Size	Dim	Big-kNN	C-Big-kNN	Oracle-kNN	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-kNN compared to Oracle-kNN in real datasets. Best performance is shown in bold-face. The speedup factor is defined as computing time of Oracle-kNN divided by the time of the slower Big-kNN method. Oracle $k = N^{0.7}$, number of subsets $s = N^{0.3}$.


Thank
you!