Today.

Modelling.

Today.

Modelling.

An Analysis of the Power of PCA.

Today.

Modelling.

An Analysis of the Power of PCA.

Musing about “heuristics” in the real world.

Two populations.

DNA data:

Two populations.

DNA data:

human1: A $\cdots$ C $\cdots$ T $\cdots$ A

Two populations.

DNA data:

human1: A $\cdots$ C $\cdots$ T $\cdots$ A
human2: C $\cdots$ C $\cdots$ A $\cdots$ T

Two populations.

DNA data:

human1: A $\cdots$ C $\cdots$ T $\cdots$ A
human2: C $\cdots$ C $\cdots$ A $\cdots$ T
human3: A $\cdots$ G $\cdots$ T $\cdots$ T

Two populations.

DNA data:

human1: A $\cdots$ C $\cdots$ T $\cdots$ A
human2: C $\cdots$ C $\cdots$ A $\cdots$ T
human3: A $\cdots$ G $\cdots$ T $\cdots$ T

Single Nucleotide Polymorphism.

Two populations.

DNA data:

human1: A $\cdots$ C $\cdots$ T $\cdots$ A
human2: C $\cdots$ C $\cdots$ A $\cdots$ T
human3: A $\cdots$ G $\cdots$ T $\cdots$ T

Single Nucleotide Polymorphism.

Same population?

Two populations.

DNA data:

human1: A $\cdots$ C $\cdots$ T $\cdots$ A
human2: C $\cdots$ C $\cdots$ A $\cdots$ T
human3: A $\cdots$ G $\cdots$ T $\cdots$ T

Single Nucleotide Polymorphism.

Same population?

Model: same popultion breeds.

Two populations.

DNA data:

human1: A $\cdots$ C $\cdots$ T $\cdots$ A
human2: C $\cdots$ C $\cdots$ A $\cdots$ T
human3: A $\cdots$ G $\cdots$ T $\cdots$ T

Single Nucleotide Polymorphism.

Same population?

Model: same populution breeds.

Population 1: snp 843: Pr[A] = .4 , Pr[T] = .6
Population 2: snp 843: Pr[A] = .6 , Pr[T] = .4

Two populations.

DNA data:

human1: A $\cdots$ C $\cdots$ T $\cdots$ A
human2: C $\cdots$ C $\cdots$ A $\cdots$ T
human3: A $\cdots$ G $\cdots$ T $\cdots$ T

Single Nucleotide Polymorphism.

Same population?

Model: same populution breeds.

Population 1: snp 843: Pr[A] = .4 , Pr[T] = .6
Population 2: snp 843: Pr[A] = .6 , Pr[T] = .4

Individual: $x_1, x_2, x_3.., x_n$.

Two populations.

DNA data:

human1: A $\cdots$ C $\cdots$ T $\cdots$ A
human2: C $\cdots$ C $\cdots$ A $\cdots$ T
human3: A $\cdots$ G $\cdots$ T $\cdots$ T

Single Nucleotide Polymorphism.

Same population?

Model: same populution breeds.

Population 1: snp 843: Pr[A] = .4 , Pr[T] = .6
Population 2: snp 843: Pr[A] = .6 , Pr[T] = .4

Individual: $x_1, x_2, x_3 .., x_n$.

Which population?

Two populations.

DNA data:

human1: A $\cdots$ C $\cdots$ T $\cdots$ A
human2: C $\cdots$ C $\cdots$ A $\cdots$ T
human3: A $\cdots$ G $\cdots$ T $\cdots$ T

Single Nucleotide Polymorphism.

Same population?

Model: same populution breeds.

Population 1: snp 843: Pr[A] = .4 , Pr[T] = .6
Population 2: snp 843: Pr[A] = .6 , Pr[T] = .4

Individual: $x_1, x_2, x_3.., x_n$.

Which population?

Comment: snps could be movie preferences, populations could be types.

Two populations.

DNA data:

human1: A $\cdots$ C $\cdots$ T $\cdots$ A
human2: C $\cdots$ C $\cdots$ A $\cdots$ T
human3: A $\cdots$ G $\cdots$ T $\cdots$ T

Single Nucleotide Polymorphism.

Same population?

Model: same populution breeds.

Population 1: snp 843: Pr[A] = .4 , Pr[T] = .6
Population 2: snp 843: Pr[A] = .6 , Pr[T] = .4

Individual: $x_1, x_2, x_3 .., x_n$.

Which population?

Comment: snps could be movie preferences, populations could be types.
E.g., republican/democrat, shopper/saver.

Which population?

Population 1: snp 843: Pr[A] = .4 , Pr[T] = .6
Population 2: snp 843: Pr[A] = .6 , Pr[T] = .4

Which population?

Population 1: snp 843: Pr[A] = .4 , Pr[T] = .6
Population 2: snp 843: Pr[A] = .6 , Pr[T] = .4

Individual: $x_1, x_2, x_3.., x_n$.

Which population?

Population 1: snp 843: Pr[A] = .4 , Pr[T] = .6
Population 2: snp 843: Pr[A] = .6 , Pr[T] = .4

Individual: $x_1, x_2, x_3 .., x_n$.

Population 1: snp i: Pr[$x_i = 1$] = $p_i^{(1)}$
Population 2: snp i: Pr[$x_i = 1$] = $p_i^{(2)}$

Which population?

Population 1: snp 843: Pr[A] = .4 , Pr[T] = .6
Population 2: snp 843: Pr[A] = .6 , Pr[T] = .4

Individual: $x_1, x_2, x_3 .., x_n$.

Population 1: snp i: Pr[$x_i = 1$] = $p_i^{(1)}$
Population 2: snp i: Pr[$x_i = 1$] = $p_i^{(2)}$

Simpler Calculation:

Which population?

Population 1: snp 843: Pr[A] = .4 , Pr[T] = .6
Population 2: snp 843: Pr[A] = .6 , Pr[T] = .4

Individual: $x_1, x_2, x_3 .., x_n$.

Population 1: snp i: Pr[$x_i = 1$] = $p_i^{(1)}$
Population 2: snp i: Pr[$x_i = 1$] = $p_i^{(2)}$

Simpler Calculation:

Population 1: Gaussion with mean $\mu_1 \in R^d$, variance σ in each dim.
Population 2: Gaussion with mean $\mu_2 \in R^d$, variance σ in each dim.

Which population?

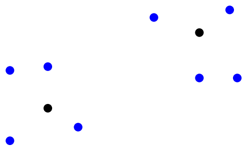
Population 1: snp 843: Pr[A] = .4 , Pr[T] = .6
Population 2: snp 843: Pr[A] = .6 , Pr[T] = .4

Individual: $x_1, x_2, x_3 .., x_n$.

Population 1: snp i: Pr[$x_i = 1$] = $p_i^{(1)}$
Population 2: snp i: Pr[$x_i = 1$] = $p_i^{(2)}$

Simpler Calculation:

Population 1: Gaussion with mean $\mu_1 \in R^d$, variance σ in each dim.
Population 2: Gaussion with mean $\mu_2 \in R^d$, variance σ in each dim.



Mixture of Gaussians

Population 1: Gaussion with mean $\mu_1 \in R^d$, std dev. σ in each dim.
Population 2: Gaussion with mean $\mu_2 \in R^d$, std dev. σ in each dim.

Mixture of Gaussians

Population 1: Gaussion with mean $\mu_1 \in R^d$, std dev. σ in each dim.
Population 2: Gaussion with mean $\mu_2 \in R^d$, std dev. σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

Mixture of Gaussians

Population 1: Gaussion with mean $\mu_1 \in R^d$, std dev. σ in each dim.
Population 2: Gaussion with mean $\mu_2 \in R^d$, std dev. σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

How many snps to collect to determine population for individual x?

Mixture of Gaussians

Population 1: Gaussion with mean $\mu_1 \in R^d$, std dev. σ in each dim.
Population 2: Gaussion with mean $\mu_2 \in R^d$, std dev. σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

How many snps to collect to determine population for individual x?
x in population 1.

Mixture of Gaussians

Population 1: Gaussion with mean $\mu_1 \in R^d$, std dev. σ in each dim.
Population 2: Gaussion with mean $\mu_2 \in R^d$, std dev. σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

How many snps to collect to determine population for individual x?
x in population 1.

$$E[(x-\mu_1)^2] = d\sigma^2$$

Mixture of Gaussians

Population 1: Gaussion with mean $\mu_1 \in R^d$, std dev. σ in each dim.
Population 2: Gaussion with mean $\mu_2 \in R^d$, std dev. σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

How many snps to collect to determine population for individual x?
x in population 1.

$$E[(x-\mu_1)^2] = d\sigma^2$$
$$E[(x-\mu_2)^2] \geq (d-1)\sigma^2 + (\mu_1-\mu_2)^2.$$

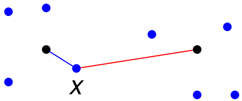
Mixture of Gaussians

Population 1: Gaussion with mean $\mu_1 \in R^d$, std dev. σ in each dim.
Population 2: Gaussion with mean $\mu_2 \in R^d$, std dev. σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

How many snps to collect to determine population for individual x?
x in population 1.

$$E[(x-\mu_1)^2] = d\sigma^2$$
$$E[(x-\mu_2)^2] \geq (d-1)\sigma^2 + (\mu_1-\mu_2)^2.$$



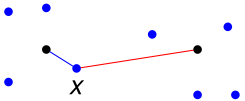
Mixture of Gaussians

Population 1: Gaussion with mean $\mu_1 \in R^d$, std dev. σ in each dim.
Population 2: Gaussion with mean $\mu_2 \in R^d$, std dev. σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

How many snps to collect to determine population for individual x?
x in population 1.

$$E[(x-\mu_1)^2] = d\sigma^2$$
$$E[(x-\mu_2)^2] \geq (d-1)\sigma^2 + (\mu_1-\mu_2)^2.$$



If $(\mu_1-\mu_2)^2 = d\varepsilon^2 >> \sigma^2$, then different.

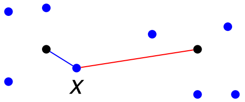
Mixture of Gaussians

Population 1: Gaussion with mean $\mu_1 \in R^d$, std dev. σ in each dim.
Population 2: Gaussion with mean $\mu_2 \in R^d$, std dev. σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

How many snps to collect to determine population for individual x?
x in population 1.

$$E[(x-\mu_1)^2] = d\sigma^2$$
$$E[(x-\mu_2)^2] \geq (d-1)\sigma^2 + (\mu_1-\mu_2)^2.$$



If $(\mu_1-\mu_2)^2 = d\varepsilon^2 >> \sigma^2$, then different.
$\rightarrow$ take $d >> \sigma^2/\varepsilon^2$

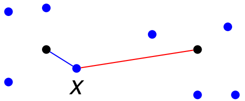
Mixture of Gaussians

Population 1: Gaussion with mean $\mu_1 \in R^d$, std dev. σ in each dim.
Population 2: Gaussion with mean $\mu_2 \in R^d$, std dev. σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

How many snps to collect to determine population for individual x?
x in population 1.

$$E[(x-\mu_1)^2] = d\sigma^2$$
$$E[(x-\mu_2)^2] \geq (d-1)\sigma^2 + (\mu_1-\mu_2)^2.$$



If $(\mu_1-\mu_2)^2 = d\varepsilon^2 >> \sigma^2$, then different.
$\rightarrow$ take $d >> \sigma^2/\varepsilon^2$

Variance of estimator?

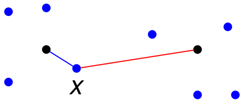
Mixture of Gaussians

Population 1: Gaussion with mean $\mu_1 \in R^d$, std dev. σ in each dim.
Population 2: Gaussion with mean $\mu_2 \in R^d$, std dev. σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

How many snps to collect to determine population for individual x?
x in population 1.

$$E[(x-\mu_1)^2] = d\sigma^2$$
$$E[(x-\mu_2)^2] \geq (d-1)\sigma^2 + (\mu_1-\mu_2)^2.$$



If $(\mu_1-\mu_2)^2 = d\varepsilon^2 >> \sigma^2$, then different.
$\rightarrow$ take $d >> \sigma^2/\varepsilon^2$

Variance of estimator?
Roughly $d\sigma^4$.

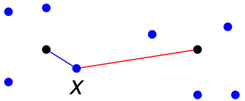
Mixture of Gaussians

Population 1: Gaussion with mean $\mu_1 \in R^d$, std dev. σ in each dim.
Population 2: Gaussion with mean $\mu_2 \in R^d$, std dev. σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

How many snps to collect to determine population for individual x?
x in population 1.

$$E[(x-\mu_1)^2] = d\sigma^2$$
$$E[(x-\mu_2)^2] \geq (d-1)\sigma^2 + (\mu_1-\mu_2)^2.$$



If $(\mu_1-\mu_2)^2 = d\varepsilon^2 >> \sigma^2$, then different.
$\rightarrow$ take $d >> \sigma^2/\varepsilon^2$

Variance of estimator?
Roughly $d\sigma^4$.
Signal is difference between expecations.

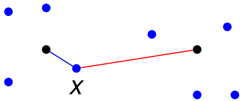
Mixture of Gaussians

Population 1: Gaussion with mean $\mu_1 \in R^d$, std dev. σ in each dim.
Population 2: Gaussion with mean $\mu_2 \in R^d$, std dev. σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

How many snps to collect to determine population for individual x?
x in population 1.

$$E[(x-\mu_1)^2] = d\sigma^2$$
$$E[(x-\mu_2)^2] \geq (d-1)\sigma^2 + (\mu_1-\mu_2)^2.$$



If $(\mu_1-\mu_2)^2 = d\varepsilon^2 >> \sigma^2$, then different.
$\rightarrow$ take $d >> \sigma^2/\varepsilon^2$

Variance of estimator?
Roughly $d\sigma^4$.
Signal is difference between expecations.
roughly $d\varepsilon^2$

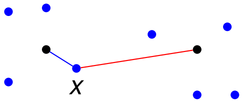
Mixture of Gaussians

Population 1: Gaussion with mean $\mu_1 \in R^d$, std dev. σ in each dim.
Population 2: Gaussion with mean $\mu_2 \in R^d$, std dev. σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

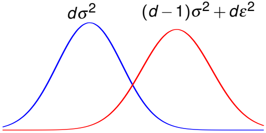
How many snps to collect to determine population for individual x?
x in population 1.

$$E[(x-\mu_1)^2] = d\sigma^2$$
$$E[(x-\mu_2)^2] \geq (d-1)\sigma^2 + (\mu_1-\mu_2)^2.$$



If $(\mu_1-\mu_2)^2 = d\varepsilon^2 >> \sigma^2$, then different.
$\rightarrow$ take $d >> \sigma^2/\varepsilon^2$

Variance of estimator?
Roughly $d\sigma^4$.
Signal is difference between expecations.
roughly $d\varepsilon^2$



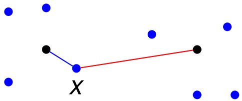
Mixture of Gaussians

Population 1: Gaussion with mean $\mu_1 \in R^d$, std dev. σ in each dim.
Population 2: Gaussion with mean $\mu_2 \in R^d$, std dev. σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

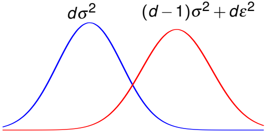
How many snps to collect to determine population for individual x?
x in population 1.

$$E[(x-\mu_1)^2] = d\sigma^2$$
$$E[(x-\mu_2)^2] \geq (d-1)\sigma^2 + (\mu_1-\mu_2)^2.$$



If $(\mu_1-\mu_2)^2 = d\varepsilon^2 >> \sigma^2$, then different.
$\rightarrow$ take $d >> \sigma^2/\varepsilon^2$

Variance of estimator?
Roughly $d\sigma^4$.
Signal is difference between expecations.
roughly $d\varepsilon^2$



Signal $>>$ Noise. $\leftrightarrow d\varepsilon^2 >> \sqrt{d}\sigma^2$.

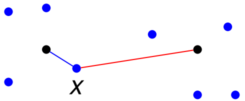
Mixture of Gaussians

Population 1: Gaussion with mean $\mu_1 \in R^d$, std dev. σ in each dim.
Population 2: Gaussion with mean $\mu_2 \in R^d$, std dev. σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

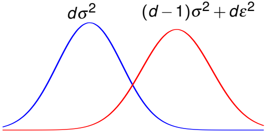
How many snps to collect to determine population for individual x?
x in population 1.

$$E[(x-\mu_1)^2] = d\sigma^2$$
$$E[(x-\mu_2)^2] \geq (d-1)\sigma^2 + (\mu_1-\mu_2)^2.$$



If $(\mu_1-\mu_2)^2 = d\varepsilon^2 >> \sigma^2$, then different.
$\rightarrow$ take $d >> \sigma^2/\varepsilon^2$

Variance of estimator?
Roughly $d\sigma^4$.
Signal is difference between expecations.
roughly $d\varepsilon^2$



Signal $>>$ Noise. $\leftrightarrow$ $d\varepsilon^2 >> \sqrt{d}\sigma^2$. $d >> \sigma^4/\varepsilon^4$ suffices!

Projection

Population 1: Gaussian with mean $\mu_1 \in R^d$, variance σ in each dim.
Population 2: Gaussian with mean $\mu_2 \in R^d$, variance σ in each dim.

Projection

Population 1: Gaussian with mean $\mu_1 \in R^d$, variance σ in each dim.
Population 2: Gaussian with mean $\mu_2 \in R^d$, variance σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

Projection

Population 1: Gaussian with mean $\mu_1 \in R^d$, variance σ in each dim.
Population 2: Gaussian with mean $\mu_2 \in R^d$, variance σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

Project x onto unit vector v in direction $\mu_2 - \mu_1$. x in population 1.

Projection

Population 1: Gaussian with mean $\mu_1 \in R^d$, variance σ in each dim.
Population 2: Gaussian with mean $\mu_2 \in R^d$, variance σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

Project x onto unit vector v in direction $\mu_2 - \mu_1$. x in population 1.

$$E[((x - \mu_1) \cdot v)^2] = \sigma^2$$

Projection

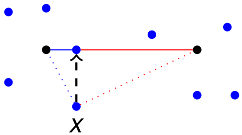
Population 1: Gaussian with mean $\mu_1 \in R^d$, variance σ in each dim.
Population 2: Gaussian with mean $\mu_2 \in R^d$, variance σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

Project x onto unit vector v in direction $\mu_2 - \mu_1$. x in population 1.

$$E[((x-\mu_1)\cdot v)^2] = \sigma^2$$

$$E[((x-\mu_2)\cdot v)^2] \geq (\mu_1 - \mu_2)^2$$



Projection

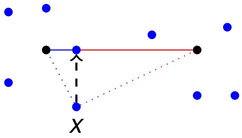
Population 1: Gaussian with mean $\mu_1 \in R^d$, variance σ in each dim.
Population 2: Gaussian with mean $\mu_2 \in R^d$, variance σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

Project x onto unit vector v in direction $\mu_2 - \mu_1$. x in population 1.

$$E[((x-\mu_1)\cdot v)^2] = \sigma^2$$

$$E[((x-\mu_2)\cdot v)^2] \geq (\mu_1-\mu_2)^2$$



Std deviation is σ^2!

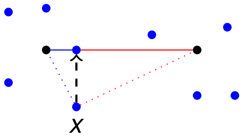
Projection

Population 1: Gaussian with mean $\mu_1 \in R^d$, variance σ in each dim.
Population 2: Gaussian with mean $\mu_2 \in R^d$, variance σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

Project x onto unit vector v in direction $\mu_2 - \mu_1$. x in population 1.

$$E[((x-\mu_1)\cdot v)^2] = \sigma^2$$
$$E[((x-\mu_2)\cdot v)^2] \geq (\mu_1-\mu_2)^2$$



Std deviation is σ^2! versus $\sqrt{d}\sigma^2$!

Projection

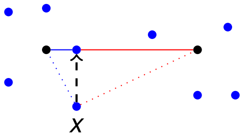
Population 1: Gaussian with mean $\mu_1 \in R^d$, variance σ in each dim.
Population 2: Gaussian with mean $\mu_2 \in R^d$, variance σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

Project x onto unit vector v in direction $\mu_2 - \mu_1$. x in population 1.

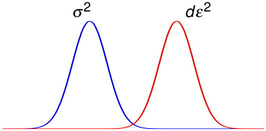
$$E[((x-\mu_1)\cdot v)^2] = \sigma^2$$

$$E[((x-\mu_2)\cdot v)^2] \geq (\mu_1 - \mu_2)^2$$



Std deviation is σ^2! versus $\sqrt{d}\sigma^2$!

No loss in signal!



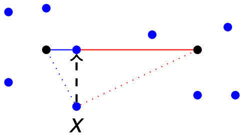
Projection

Population 1: Gaussian with mean $\mu_1 \in R^d$, variance σ in each dim.
Population 2: Gaussian with mean $\mu_2 \in R^d$, variance σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

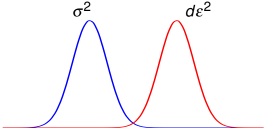
Project x onto unit vector v in direction $\mu_2 - \mu_1$. x in population 1.

$$E[((x-\mu_1)\cdot v)^2] = \sigma^2$$
$$E[((x-\mu_2)\cdot v)^2] \geq (\mu_1-\mu_2)^2$$



Std deviation is σ^2! versus $\sqrt{d}\sigma^2$!

No loss in signal!



$$d\varepsilon^2 >> \sigma^2.$$

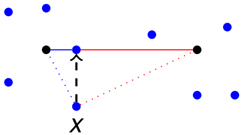
Projection

Population 1: Gaussian with mean $\mu_1 \in R^d$, variance σ in each dim.
Population 2: Gaussian with mean $\mu_2 \in R^d$, variance σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

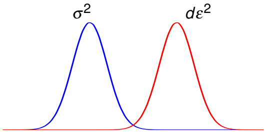
Project x onto unit vector v in direction $\mu_2 - \mu_1$. x in population 1.

$$E[((x-\mu_1)\cdot v)^2] = \sigma^2$$
$$E[((x-\mu_2)\cdot v)^2] \geq (\mu_1-\mu_2)^2$$



Std deviation is σ^2! versus $\sqrt{d}\sigma^2$!

No loss in signal!



$d\varepsilon^2 >> \sigma^2$. $\quad \rightarrow d >> \sigma^2/\varepsilon^2$

Projection

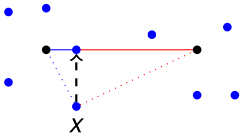
Population 1: Gaussian with mean $\mu_1 \in R^d$, variance σ in each dim.
Population 2: Gaussian with mean $\mu_2 \in R^d$, variance σ in each dim.

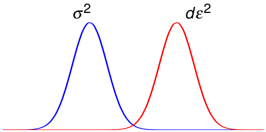
Difference between humans σ per snp.
Difference between populations ε per snp.

Project x onto unit vector v in direction $\mu_2 - \mu_1$. x in population 1.

$$E[((x-\mu_1)\cdot v)^2] = \sigma^2$$

$$E[((x-\mu_2)\cdot v)^2] \geq (\mu_1-\mu_2)^2$$





Std deviation is σ^2! versus $\sqrt{d}\sigma^2$!

No loss in signal!

$d\varepsilon^2 >> \sigma^2$. $\rightarrow d >> \sigma^2/\varepsilon^2$

Versus $d >> \sigma^4/\varepsilon^4$.

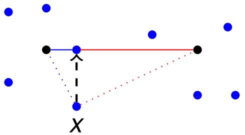
Projection

Population 1: Gaussian with mean $\mu_1 \in R^d$, variance σ in each dim.
Population 2: Gaussian with mean $\mu_2 \in R^d$, variance σ in each dim.

Difference between humans σ per snp.
Difference between populations ε per snp.

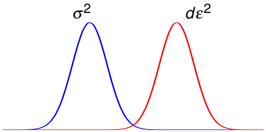
Project x onto unit vector v in direction $\mu_2 - \mu_1$. x in population 1.

$$E[((x-\mu_1)\cdot v)^2] = \sigma^2$$
$$E[((x-\mu_2)\cdot v)^2] \geq (\mu_1 - \mu_2)^2$$



Std deviation is σ^2! versus $\sqrt{d}\sigma^2$!

No loss in signal!



$d\varepsilon^2 >> \sigma^2$. $\quad \rightarrow d >> \sigma^2/\varepsilon^2$

Versus $d >> \sigma^4/\varepsilon^4$.

A quadratic difference in amount of data!

Don’t know much about...

Don’t know μ_1 or μ_2?

Don't know much about...

Don't know μ_1 or μ_2? Uh oh!

Without knowing the means?

Sample of n people.

Without knowing the means?

Sample of n people.

- Some (say half) from population 1,

Without knowing the means?

Sample of n people.

Some (say half) from population 1,

some from population 2.

Without knowing the means?

Sample of n people.

Some (say half) from population 1,

some from population 2.

Which are which?

Without knowing the means?

Sample of n people.

Some (say half) from population 1,

some from population 2.

Which are which?

Near Neighbors Approach

Without knowing the means?

Sample of n people.

Some (say half) from population 1,

some from population 2.

Which are which?

Near Neighbors Approach

Compute Euclidean distance squared.

Without knowing the means?

Sample of n people.
Some (say half) from population 1,
some from population 2.

Which are which?

Near Neighbors Approach

Compute Euclidean distance squared.   Cluster using threshold.

Without knowing the means?

Sample of n people.

Some (say half) from population 1,

some from population 2.

Which are which?

Near Neighbors Approach

Compute Euclidean distance squared.   Cluster using threshold.

Signal $E[d(x_1, x_2)] - E[d(x_1, y_1)]$

Without knowing the means?

Sample of n people.

Some (say half) from population 1,

some from population 2.

Which are which?

Near Neighbors Approach

Compute Euclidean distance squared. Cluster using threshold.

Signal $E[d(x_1, x_2)] - E[d(x_1, y_1)]$

should be larger than noise in $d(x, y)$

Without knowing the means?

Sample of n people.

Some (say half) from population 1,

some from population 2.

Which are which?

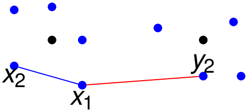
Near Neighbors Approach

Compute Euclidean distance squared. Cluster using threshold.

Signal $E[d(x_1, x_2)] - E[d(x_1, y_1)]$

should be larger than noise in $d(x, y)$

Where x's from one population, y's from other.



Without knowing the means?

Sample of n people.

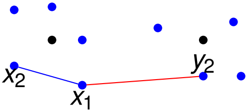
Some (say half) from population 1,

some from population 2.

Which are which?

Near Neighbors Approach

Compute Euclidean distance squared. Cluster using threshold.



Signal $E[d(x_1, x_2)] - E[d(x_1, y_1)]$
should be larger than noise in $d(x, y)$
Where x's from one population, y's from other.

Signal is proportional $d\varepsilon^2$.

Without knowing the means?

Sample of n people.

Some (say half) from population 1,

some from population 2.

Which are which?

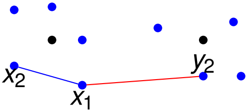
Near Neighbors Approach

Compute Euclidean distance squared. Cluster using threshold.

Signal $E[d(x_1, x_2)] - E[d(x_1, y_1)]$

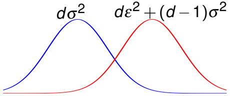
should be larger than noise in $d(x, y)$

Where x's from one population, y's from other.



Signal is proportional $d\varepsilon^2$.

Noise is proportional to $\sqrt{d}\sigma^2$.



Without knowing the means?

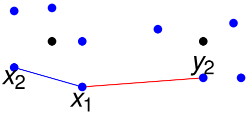
Sample of n people.
Some (say half) from population 1,
some from population 2.

Which are which?

Near Neighbors Approach

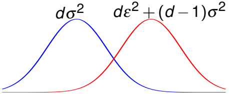
Compute Euclidean distance squared. Cluster using threshold.

Signal $E[d(x_1,x_2)] - E[d(x_1,y_1)]$
should be larger than noise in $d(x,y)$
Where x's from one population, y's from other.



Signal is proportional $d\varepsilon^2$.

Noise is proportional to $\sqrt{d}\sigma^2$.



$d >> \sigma^4/\varepsilon^4$ $\rightarrow$ same type people closer to each other.

Without knowing the means?

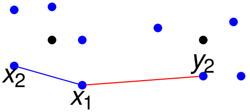
Sample of n people.
Some (say half) from population 1,
some from population 2.

Which are which?

Near Neighbors Approach

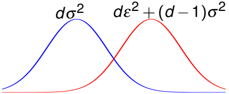
Compute Euclidean distance squared. Cluster using threshold.

Signal $E[d(x_1,x_2)] - E[d(x_1,y_1)]$
should be larger than noise in $d(x,y)$
Where x's from one population, y's from other.



Signal is proportional $d\varepsilon^2$.

Noise is proportional to $\sqrt{d}\sigma^2$.



$d >> \sigma^4/\varepsilon^4$ $\rightarrow$ same type people closer to each other.

$d >> (\sigma^4/\varepsilon^4)\log n$ suffices for threshold clustering.

Without knowing the means?

Sample of n people.

Some (say half) from population 1,

some from population 2.

Which are which?

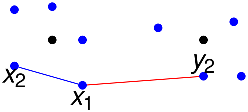
Near Neighbors Approach

Compute Euclidean distance squared. Cluster using threshold.

Signal $E[d(x_1,x_2)] - E[d(x_1,y_1)]$

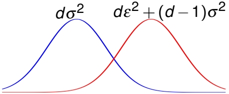
should be larger than noise in $d(x,y)$

Where x's from one population, y's from other.



Signal is proportional $d\varepsilon^2$.

Noise is proportional to $\sqrt{d}\sigma^2$.



$d >> \sigma^4/\varepsilon^4$ $\rightarrow$ same type people closer to each other.

$d >> (\sigma^4/\varepsilon^4)\log n$ suffices for threshold clustering.

$\log n$ factor for union bound over $\binom{n}{2}$ pairs.

Without knowing the means?

Sample of n people.

Some (say half) from population 1,

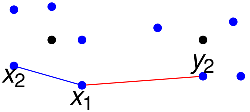
some from population 2.

Which are which?

Near Neighbors Approach

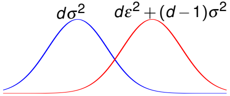
Compute Euclidean distance squared. Cluster using threshold.

Signal $E[d(x_1, x_2)] - E[d(x_1, y_1)]$
should be larger than noise in $d(x, y)$
Where x's from one population, y's from other.



Signal is proportional $d\varepsilon^2$.

Noise is proportional to $\sqrt{d}\sigma^2$.



$d >> \sigma^4/\varepsilon^4$ $\rightarrow$ same type people closer to each other.

$d >> (\sigma^4/\varepsilon^4)\log n$ suffices for threshold clustering.

$\log n$ factor for union bound over $\binom{n}{2}$ pairs.

Best one can do?

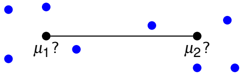
Principal components analysis.

Remember Projection!

Principal components analysis.

Remember Projection!

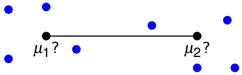
Still don't know μ_1 or μ_2?



Principal components analysis.

Remember Projection!

Still don't know μ_1 or μ_2?

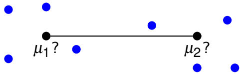


Principal component analysis(PCA):

Principal components analysis.

Remember Projection!

Still don't know μ_1 or μ_2?



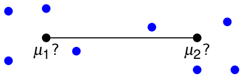
Principal component analysis(PCA):

Find direction, v, of maximum variance.

Principal components analysis.

Remember Projection!

Still don't know μ_1 or μ_2?



Principal component analysis(PCA):

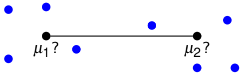
Find direction, v, of maximum variance.

Maximize $\sum(x \cdot v)^2$ (zero center the points)

Principal components analysis.

Remember Projection!

Still don't know μ_1 or μ_2?



Principal component analysis(PCA):

Find direction, v, of maximum variance.

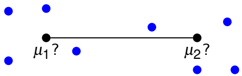
Maximize $\sum(x \cdot v)^2$ (zero center the points)

Recall: $(x \cdot v)^2$ determines population well.

Principal components analysis.

Remember Projection!

Still don't know μ_1 or μ_2?



Principal component analysis(PCA):

Find direction, v, of maximum variance.

Maximize $\sum(x \cdot v)^2$ (zero center the points)

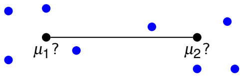
Recall: $(x \cdot v)^2$ determines population well.

Typical direction variance. $n\sigma^2$.

Principal components analysis.

Remember Projection!

Still don't know μ_1 or μ_2?



Principal component analysis(PCA):

Find direction, v, of maximum variance.

Maximize $\sum(x \cdot v)^2$ (zero center the points)

Recall: $(x \cdot v)^2$ determines population well.

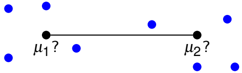
Typical direction variance. $n\sigma^2$.

Direction along $\mu_1 - \mu_2$,

Principal components analysis.

Remember Projection!

Still don't know μ_1 or μ_2?



Principal component analysis(PCA):

Find direction, v, of maximum variance.

Maximize $\sum(x \cdot v)^2$ (zero center the points)

Recall: $(x \cdot v)^2$ determines population well.

Typical direction variance. $n\sigma^2$.

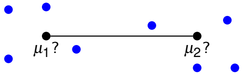
Direction along $\mu_1 - \mu_2$,

$\propto n(\mu_1 - \mu_2)^2$.

Principal components analysis.

Remember Projection!

Still don't know μ_1 or μ_2?



Principal component analysis(PCA):

Find direction, v, of maximum variance.

Maximize $\sum(x \cdot v)^2$ (zero center the points)

Recall: $(x \cdot v)^2$ determines population well.

Typical direction variance. $n\sigma^2$.

Direction along $\mu_1 - \mu_2$,

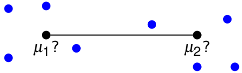
$\propto n(\mu_1 - \mu_2)^2$.

$\propto nd\varepsilon^2$.

Principal components analysis.

Remember Projection!

Still don't know μ_1 or μ_2?



Principal component analysis(PCA):

Find direction, v, of maximum variance.

Maximize $\sum (x \cdot v)^2$ (zero center the points)

Recall: $(x \cdot v)^2$ determines population well.

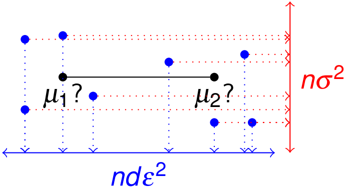
Typical direction variance. $n\sigma^2$.

Direction along $\mu_1 - \mu_2$,

$\propto n(\mu_1 - \mu_2)^2$.

$\propto nd\varepsilon^2$.

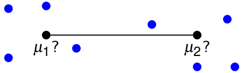
Need $nd\varepsilon^2 >> n\sigma^2$.



Principal components analysis.

Remember Projection!

Still don't know μ_1 or μ_2?



Principal component analysis(PCA):

Find direction, v, of maximum variance.

Maximize $\sum(x \cdot v)^2$ (zero center the points)

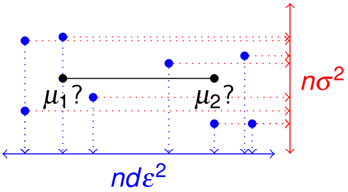
Recall: $(x \cdot v)^2$ determines population well.

Typical direction variance. $n\sigma^2$.

Direction along $\mu_1 - \mu_2$,

$\propto n(\mu_1 - \mu_2)^2$.

$\propto nd\varepsilon^2$.



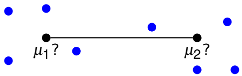
Need $nd\varepsilon^2 >> n\sigma^2$.

Need $d\varepsilon^2 >> \sigma^2 \equiv d >> \sigma^2/\varepsilon^2$ at least.

Principal components analysis.

Remember Projection!

Still don't know μ_1 or μ_2?



Principal component analysis(PCA):

Find direction, v, of maximum variance.

Maximize $\sum(x \cdot v)^2$ (zero center the points)

Recall: $(x \cdot v)^2$ determines population well.

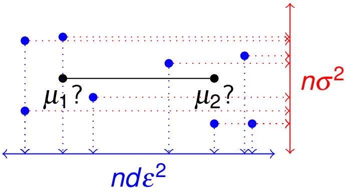
Typical direction variance. $n\sigma^2$.

Direction along $\mu_1 - \mu_2$,

$\propto n(\mu_1 - \mu_2)^2$.

$\propto nd\varepsilon^2$.

Need $nd\varepsilon^2 >> n\sigma^2$.

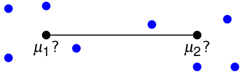


Need $d\varepsilon^2 >> \sigma^2 \equiv d >> \sigma^2/\varepsilon^2$ at least. Just signal difference here!

Principal components analysis.

Remember Projection!

Still don't know μ_1 or μ_2?



Principal component analysis(PCA):

Find direction, v, of maximum variance.

Maximize $\sum(x \cdot v)^2$ (zero center the points)

Recall: $(x \cdot v)^2$ determines population well.

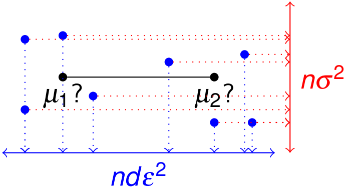
Typical direction variance. $n\sigma^2$.

Direction along $\mu_1 - \mu_2$,

$\propto n(\mu_1 - \mu_2)^2$.

$\propto nd\varepsilon^2$.

Need $nd\varepsilon^2 >> n\sigma^2$.



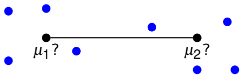
Need $d\varepsilon^2 >> \sigma^2 \equiv d >> \sigma^2/\varepsilon^2$ at least. Just signal difference here!

$nd\varepsilon^2 >> n\sigma^2$.

Principal components analysis.

Remember Projection!

Still don't know μ_1 or μ_2?



Principal component analysis(PCA):

Find direction, v, of maximum variance.

Maximize $\sum(x \cdot v)^2$ (zero center the points)

Recall: $(x \cdot v)^2$ determines population well.

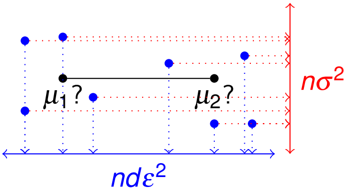
Typical direction variance. $n\sigma^2$.

Direction along $\mu_1 - \mu_2$,

$\propto n(\mu_1 - \mu_2)^2$.

$\propto nd\varepsilon^2$.

Need $nd\varepsilon^2 >> n\sigma^2$.



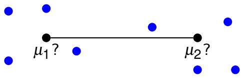
Need $d\varepsilon^2 >> \sigma^2 \equiv d >> \sigma^2/\varepsilon^2$ at least. Just signal difference here!

$nd\varepsilon^2 >> n\sigma^2$. Direction $\mu_2 - \mu_1$, best in expectation...but..

Principal components analysis.

Remember Projection!

Still don't know μ_1 or μ_2?



Principal component analysis(PCA):

Find direction, v, of maximum variance.

Maximize $\sum(x \cdot v)^2$ (zero center the points)

Recall: $(x \cdot v)^2$ determines population well.

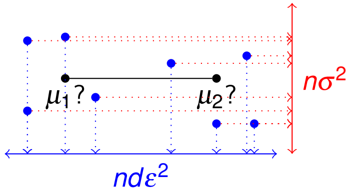
Typical direction variance. $n\sigma^2$.

Direction along $\mu_1 - \mu_2$,

$\propto n(\mu_1 - \mu_2)^2$.

$\propto nd\varepsilon^2$.

Need $nd\varepsilon^2 >> n\sigma^2$.



Need $d\varepsilon^2 >> \sigma^2 \equiv d >> \sigma^2/\varepsilon^2$ at least. Just signal difference here!

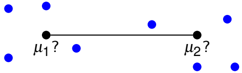
$nd\varepsilon^2 >> n\sigma^2$. Direction $\mu_2 - \mu_1$, best in expectation...but..

When will PCA pick correct direction with good probability?

Principal components analysis.

Remember Projection!

Still don't know μ_1 or μ_2?



Principal component analysis(PCA):

Find direction, v, of maximum variance.

Maximize $\sum(x \cdot v)^2$ (zero center the points)

Recall: $(x \cdot v)^2$ determines population well.

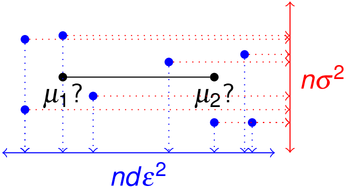
Typical direction variance. $n\sigma^2$.

Direction along $\mu_1 - \mu_2$,

$\propto n(\mu_1 - \mu_2)^2$.

$\propto nd\varepsilon^2$.

Need $nd\varepsilon^2 >> n\sigma^2$.



Need $d\varepsilon^2 >> \sigma^2 \equiv d >> \sigma^2/\varepsilon^2$ at least. Just signal difference here!

$nd\varepsilon^2 >> n\sigma^2$. Direction $\mu_2 - \mu_1$, best in expectation...but..

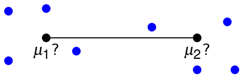
When will PCA pick correct direction with good probability?

Union bound?

Principal components analysis.

Remember Projection!

Still don't know μ_1 or μ_2?



Principal component analysis(PCA):

Find direction, v, of maximum variance.

Maximize $\sum(x \cdot v)^2$ (zero center the points)

Recall: $(x \cdot v)^2$ determines population well.

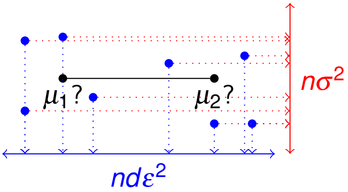
Typical direction variance. $n\sigma^2$.

Direction along $\mu_1 - \mu_2$,

$\propto n(\mu_1 - \mu_2)^2$.

$\propto nd\varepsilon^2$.

Need $nd\varepsilon^2 >> n\sigma^2$.



Need $d\varepsilon^2 >> \sigma^2 \equiv d >> \sigma^2/\varepsilon^2$ at least. Just signal difference here!

$nd\varepsilon^2 >> n\sigma^2$. Direction $\mu_2 - \mu_1$, best in expectation...but..

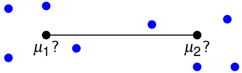
When will PCA pick correct direction with good probability?

Union bound? How many directions?

Principal components analysis.

Remember Projection!

Still don't know μ_1 or μ_2?



Principal component analysis(PCA):

Find direction, v, of maximum variance.

Maximize $\sum(x \cdot v)^2$ (zero center the points)

Recall: $(x \cdot v)^2$ determines population well.

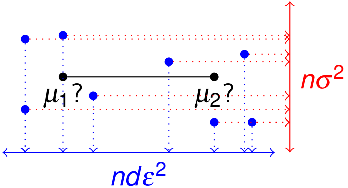
Typical direction variance. $n\sigma^2$.

Direction along $\mu_1 - \mu_2$,

$\propto n(\mu_1 - \mu_2)^2$.

$\propto nd\varepsilon^2$.

Need $nd\varepsilon^2 >> n\sigma^2$.



Need $d\varepsilon^2 >> \sigma^2 \equiv d >> \sigma^2/\varepsilon^2$ at least. Just signal difference here!

$nd\varepsilon^2 >> n\sigma^2$. Direction $\mu_2 - \mu_1$, best in expectation...but..

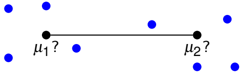
When will PCA pick correct direction with good probability?

Union bound? How many directions? Infinity

Principal components analysis.

Remember Projection!

Still don't know μ_1 or μ_2?



Principal component analysis(PCA):

Find direction, v, of maximum variance.

Maximize $\sum(x \cdot v)^2$ (zero center the points)

Recall: $(x \cdot v)^2$ determines population well.

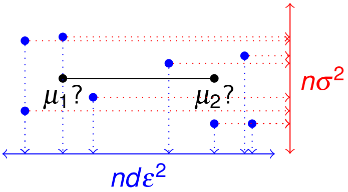
Typical direction variance. $n\sigma^2$.

Direction along $\mu_1 - \mu_2$,

$\propto n(\mu_1 - \mu_2)^2$.

$\propto nd\varepsilon^2$.

Need $nd\varepsilon^2 >> n\sigma^2$.



Need $d\varepsilon^2 >> \sigma^2 \equiv d >> \sigma^2/\varepsilon^2$ at least. Just signal difference here!

$nd\varepsilon^2 >> n\sigma^2$. Direction $\mu_2 - \mu_1$, best in expectation...but..

When will PCA pick correct direction with good probability?

Union bound? How many directions? Infinity and beyond!

Nets

“δ - Net”.

Nets

"δ - Net".

Set $\mathscr{D}$ of directions

Nets

"δ - Net".

Set $\mathscr{D}$ of directions

where all others, v, are close to $x \in \mathscr{D}$.

Nets

“δ - Net”.

- Set $\mathscr{D}$ of directions
  - where all others, v, are close to $x \in \mathscr{D}$.
    - $x \cdot v \geq 1 - \delta$.

Nets

"δ - Net".

Set $\mathscr{D}$ of directions

where all others, v, are close to $x \in \mathscr{D}$.

$x \cdot v \geq 1 - \delta$.

δ- Net:

Nets

"δ - Net".

Set $\mathscr{D}$ of directions

where all others, v, are close to $x \in \mathscr{D}$.

$x \cdot v \geq 1 - \delta$.

δ- Net:

$[\cdots, i\delta/\sqrt{d}, \cdots]$, with $i \in [-\sqrt{d}/\delta, \sqrt{d}/\delta]$.

Nets

"δ - Net".

Set $\mathscr{D}$ of directions

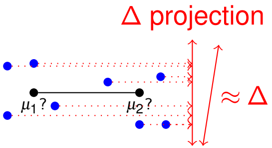
where all others, v, are close to $x \in \mathscr{D}$.

$x \cdot v \geq 1 - \delta$.

δ- Net:

$[\cdots, i\delta/\sqrt{d}, \cdots]$, with $i \in [-\sqrt{d}/\delta, \sqrt{d}/\delta]$.

Total of $N \propto \left(\frac{d}{\delta}\right)^{O(d)}$ vectors in net.



Nets

"δ - Net".

Set $\mathscr{D}$ of directions

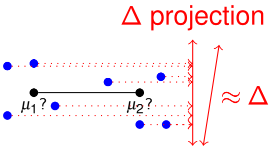
where all others, v, are close to $x \in \mathscr{D}$.

$x \cdot v \geq 1 - \delta$.

δ- Net:

$[\cdots, i\delta/\sqrt{d}, \cdots]$, with $i \in [-\sqrt{d}/\delta, \sqrt{d}/\delta]$.

Total of $N \propto \left(\frac{d}{\delta}\right)^{O(d)}$ vectors in net.



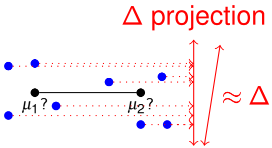
Signal $>>$ Noise times $\sqrt{\log N} = O(\sqrt{d \log \frac{d}{\delta}})$ to isolate direction.

Nets

"δ - Net".
Set $\mathscr{D}$ of directions
where all others, v, are close to $x \in \mathscr{D}$.
$x \cdot v \geq 1 - \delta$.

δ- Net:
$[\cdots, i\delta/\sqrt{d}, \cdots]$, with $i \in [-\sqrt{d}/\delta, \sqrt{d}/\delta]$.

Total of $N \propto \left(\frac{d}{\delta}\right)^{O(d)}$ vectors in net.



Signal $>>$ Noise times $\sqrt{\log N} = O(\sqrt{d \log \frac{d}{\delta}})$ to isolate direction.
Recall $e^{-t^2/2}$ Gaussian behavior.

Nets

"δ - Net".

Set $\mathscr{D}$ of directions

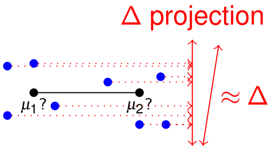
where all others, v, are close to $x \in \mathscr{D}$.

$x \cdot v \geq 1 - \delta$.

δ- Net:

$[\cdots, i\delta/\sqrt{d}, \cdots]$, with $i \in [-\sqrt{d}/\delta, \sqrt{d}/\delta]$.

Total of $N \propto \left(\frac{d}{\delta}\right)^{O(d)}$ vectors in net.



Signal $>>$ Noise times $\sqrt{\log N} = O(\sqrt{d \log \frac{d}{\delta}})$ to isolate direction.

Recall $e^{-t^2/2}$ Gaussian behavior.

$t = \sqrt{\log N}$ to union bound over vectors in net.

Nets

"δ - Net".

Set $\mathscr{D}$ of directions

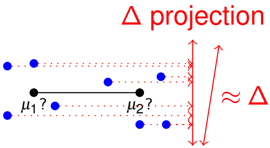
where all others, v, are close to $x \in \mathscr{D}$.

$x \cdot v \geq 1 - \delta$.

δ- Net:

$[\cdots, i\delta/\sqrt{d}, \cdots]$, with $i \in [-\sqrt{d}/\delta, \sqrt{d}/\delta]$.

Total of $N \propto \left(\frac{d}{\delta}\right)^{O(d)}$ vectors in net.



Signal >> Noise times $\sqrt{\log N} = O(\sqrt{d \log \frac{d}{\delta}})$ to isolate direction.

Recall $e^{-t^2/2}$ Gaussian behavior.

$t = \sqrt{\log N}$ to union bound over vectors in net.

Signal (exp. projection): $\propto nd\varepsilon^2$.

Nets

"δ - Net".

Set $\mathscr{D}$ of directions

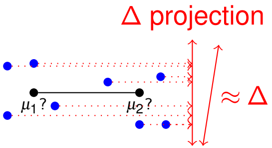
where all others, v, are close to $x \in \mathscr{D}$.

$x \cdot v \geq 1 - \delta$.

δ- Net:

$[\cdots, i\delta/\sqrt{d}, \cdots]$, with $i \in [-\sqrt{d}/\delta, \sqrt{d}/\delta]$.

Total of $N \propto \left(\frac{d}{\delta}\right)^{O(d)}$ vectors in net.



Signal $>>$ Noise times $\sqrt{\log N} = O(\sqrt{d \log \frac{d}{\delta}})$ to isolate direction.

Recall $e^{-t^2/2}$ Gaussian behavior.

$t = \sqrt{\log N}$ to union bound over vectors in net.

Signal (exp. projection): $\propto nd\varepsilon^2$.

Noise (std dev.): $\sqrt{n}\sigma^2$.

Nets

"δ - Net".

Set $\mathscr{D}$ of directions

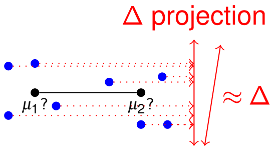
where all others, v, are close to $x \in \mathscr{D}$.

$x \cdot v \geq 1 - \delta$.

δ- Net:

$[\cdots, i\delta/\sqrt{d}, \cdots]$, with $i \in [-\sqrt{d}/\delta, \sqrt{d}/\delta]$.

Total of $N \propto \left(\frac{d}{\delta}\right)^{O(d)}$ vectors in net.



Signal >> Noise times $\sqrt{\log N} = O(\sqrt{d \log \frac{d}{\delta}})$ to isolate direction.

Recall $e^{-t^2/2}$ Gaussian behavior.

$t = \sqrt{\log N}$ to union bound over vectors in net.

Signal (exp. projection): $\propto nd\varepsilon^2$.

Noise (std dev.): $\sqrt{n}\sigma^2$. union bound: $\sqrt{nd \log d} \times \sigma^2$

Nets

"δ - Net".

Set $\mathscr{D}$ of directions

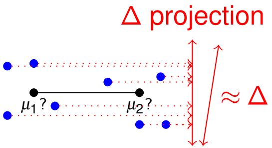
where all others, v, are close to $x \in \mathscr{D}$.

$x \cdot v \geq 1 - \delta$.

δ- Net:

$[\cdots, i\delta/\sqrt{d}, \cdots]$, with $i \in [-\sqrt{d}/\delta, \sqrt{d}/\delta]$.

Total of $N \propto \left(\frac{d}{\delta}\right)^{O(d)}$ vectors in net.



Signal >> Noise times $\sqrt{\log N} = O(\sqrt{d \log \frac{d}{\delta}})$ to isolate direction.

Recall $e^{-t^2/2}$ Gaussian behavior.

$t = \sqrt{\log N}$ to union bound over vectors in net.

Signal (exp. projection): $\propto nd\varepsilon^2$.

Noise (std dev.): $\sqrt{n}\sigma^2$. union bound: $\sqrt{nd \log d} \times \sigma^2$

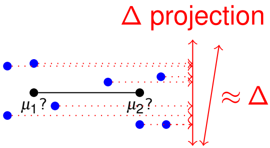
$nd >> (\sigma^4/\varepsilon^4) \log d$ and $d >> \sigma^2/\varepsilon^2$ works.

Nets

"δ - Net".
Set $\mathscr{D}$ of directions
where all others, v, are close to $x \in \mathscr{D}$.
$x \cdot v \geq 1 - \delta$.

δ- Net:
$[\cdots, i\delta/\sqrt{d}, \cdots]$, with $i \in [-\sqrt{d}/\delta, \sqrt{d}/\delta]$.

Total of $N \propto \left(\frac{d}{\delta}\right)^{O(d)}$ vectors in net.



Signal $>>$ Noise times $\sqrt{\log N} = O(\sqrt{d \log \frac{d}{\delta}})$ to isolate direction.
Recall $e^{-t^2/2}$ Gaussian behavior.
$t = \sqrt{\log N}$ to union bound over vectors in net.

Signal (exp. projection): $\propto nd\varepsilon^2$.
Noise (std dev.): $\sqrt{n}\sigma^2$. union bound: $\sqrt{nd \log d} \times \sigma^2$

$nd >> (\sigma^4/\varepsilon^4) \log d$ and $d >> \sigma^2/\varepsilon^2$ works.

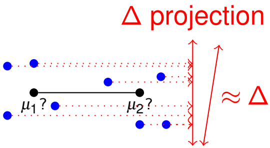
Nearest neighbor works with very high $d > \sigma^4/\varepsilon^4$.

Nets

"δ - Net".
Set $\mathscr{D}$ of directions
where all others, v, are close to $x \in \mathscr{D}$.
$x \cdot v \geq 1 - \delta$.

δ- Net:
$[\cdots, i\delta/\sqrt{d}, \cdots]$, with $i \in [-\sqrt{d}/\delta, \sqrt{d}/\delta]$.

Total of $N \propto \left(\frac{d}{\delta}\right)^{O(d)}$ vectors in net.



Signal $>>$ Noise times $\sqrt{\log N} = O(\sqrt{d \log \frac{d}{\delta}})$ to isolate direction.
Recall $e^{-t^2/2}$ Gaussian behavior.
$t = \sqrt{\log N}$ to union bound over vectors in net.

Signal (exp. projection): $\propto nd\varepsilon^2$.
Noise (std dev.): $\sqrt{n}\sigma^2$. union bound: $\sqrt{nd \log d} \times \sigma^2$

$nd >> (\sigma^4/\varepsilon^4) \log d$ and $d >> \sigma^2/\varepsilon^2$ works.

Nearest neighbor works with very high $d > \sigma^4/\varepsilon^4$.

PCA reduces d to "knowing centers" case, with extra sample points.

PCA calculation.

Matrix A where rows are points.

PCA calculation.

Matrix A where rows are points.

First eigenvector of $B = A^T A$ is maximum variance direction.

PCA calculation.

Matrix A where rows are points.

First eigenvector of $B = A^T A$ is maximum variance direction.

Av are projections onto v.

PCA calculation.

Matrix A where rows are points.

First eigenvector of $B = A^T A$ is maximum variance direction.

Av are projections onto v.
$vBv = (vA)^T(Av)$ is $\sum_x (x \cdot v)^2$.

PCA calculation.

Matrix A where rows are points.

First eigenvector of $B = A^T A$ is maximum variance direction.

Av are projections onto v.
$vBv = (vA)^T (Av)$ is $\sum_x (x \cdot v)^2$.

PCA calculation.

Matrix A where rows are points.

First eigenvector of $B = A^T A$ is maximum variance direction.

Av are projections onto v.
$vBv = (vA)^T(Av)$ is $\sum_x (x \cdot v)^2$.

First eigenvector, v, of B maximizes $x^T Bx$.

PCA calculation.

Matrix A where rows are points.

First eigenvector of $B = A^T A$ is maximum variance direction.

Av are projections onto v.
$vBv = (vA)^T(Av)$ is $\sum_x (x \cdot v)^2$.

First eigenvector, v, of B maximizes $x^T Bx$.

$Bv = \lambda v$ for maximum λ.

PCA calculation.

Matrix A where rows are points.

First eigenvector of $B = A^T A$ is maximum variance direction.

Av are projections onto v.
$vBv = (vA)^T(Av)$ is $\sum_x (x \cdot v)^2$.

First eigenvector, v, of B maximizes $x^T Bx$.

$Bv = \lambda v$ for maximum λ.
$\rightarrow vBv = \lambda$ for unit v.

PCA calculation.

Matrix A where rows are points.

First eigenvector of $B = A^T A$ is maximum variance direction.

Av are projections onto v.
$vBv = (vA)^T(Av)$ is $\sum_x (x \cdot v)^2$.

First eigenvector, v, of B maximizes $x^T Bx$.

$Bv = \lambda v$ for maximum λ.
$\rightarrow vBv = \lambda$ for unit v.

PCA calculation.

Matrix A where rows are points.

First eigenvector of $B = A^T A$ is maximum variance direction.

Av are projections onto v.
$vBv = (vA)^T(Av)$ is $\sum_x (x \cdot v)^2$.

First eigenvector, v, of B maximizes $x^T Bx$.

$Bv = \lambda v$ for maximum λ.
$\rightarrow vBv = \lambda$ for unit v.

Eigenvectors of $A^T A$ form orthonormal basis.

PCA calculation.

Matrix A where rows are points.

First eigenvector of $B = A^T A$ is maximum variance direction.

Av are projections onto v.
$vBv = (vA)^T(Av)$ is $\sum_x (x \cdot v)^2$.

First eigenvector, v, of B maximizes $x^T Bx$.

$Bv = \lambda v$ for maximum λ.
$\rightarrow vBv = \lambda$ for unit v.

Eigenvectors of $A^T A$ form orthonormal basis.

Intuition: $\max \frac{1}{2} v^T Bv + \lambda(1 - |v|^2)$

PCA calculation.

Matrix A where rows are points.

First eigenvector of $B = A^T A$ is maximum variance direction.

Av are projections onto v.
$vBv = (vA)^T(Av)$ is $\sum_x (x \cdot v)^2$.

First eigenvector, v, of B maximizes $x^T Bx$.

$Bv = \lambda v$ for maximum λ.
$\rightarrow vBv = \lambda$ for unit v.

Eigenvectors of $A^T A$ form orthonormal basis.

Intuition: $\max \frac{1}{2} v^T Bv + \lambda(1 - |v|^2)$
$\rightarrow Bv = \lambda v$.

PCA calculation.

Matrix A where rows are points.

First eigenvector of $B = A^T A$ is maximum variance direction.

Av are projections onto v.
$vBv = (vA)^T(Av)$ is $\sum_x (x \cdot v)^2$.

First eigenvector, v, of B maximizes $x^T Bx$.

$Bv = \lambda v$ for maximum λ.
$\rightarrow vBv = \lambda$ for unit v.

Eigenvectors of $A^T A$ form orthonormal basis.

Intuition: $\max \frac{1}{2} v^T Bv + \lambda(1 - |v|^2)$
$\rightarrow Bv = \lambda v$.

Any other vector $av + (1-a)x$, $x \cdot v = 0$

PCA calculation.

Matrix A where rows are points.

First eigenvector of $B = A^T A$ is maximum variance direction.

Av are projections onto v.
$vBv = (vA)^T(Av)$ is $\sum_x (x \cdot v)^2$.

First eigenvector, v, of B maximizes $x^T Bx$.

$Bv = \lambda v$ for maximum λ.
$\rightarrow vBv = \lambda$ for unit v.

Eigenvectors of $A^T A$ form orthonormal basis.

Intuition: $\max \frac{1}{2} v^T Bv + \lambda(1 - |v|^2)$
$\rightarrow Bv = \lambda v$.

Any other vector $av + (1-a)x$, $x \cdot v = 0$
x is composed of possibly smaller eigenvalue vectors.

PCA calculation.

Matrix A where rows are points.

First eigenvector of $B = A^T A$ is maximum variance direction.

Av are projections onto v.
$vBv = (vA)^T(Av)$ is $\sum_x (x \cdot v)^2$.

First eigenvector, v, of B maximizes $x^T Bx$.

$Bv = \lambda v$ for maximum λ.
$\rightarrow vBv = \lambda$ for unit v.

Eigenvectors of $A^T A$ form orthonormal basis.

Intuition: $\max \frac{1}{2} v^T Bv + \lambda(1 - |v|^2)$
$\rightarrow Bv = \lambda v$.

Any other vector $av + (1-a)x$, $x \cdot v = 0$
x is composed of possibly smaller eigenvalue vectors.

$\rightarrow vBv \geq (av + (1-a)x)^T B(av + (1-a)x) = a^2 v^T Bv + (1-a)^2 x^T Bx$

PCA calculation.

Matrix A where rows are points.

First eigenvector of $B = A^T A$ is maximum variance direction.

Av are projections onto v.
$vBv = (vA)^T(Av)$ is $\sum_x (x \cdot v)^2$.

First eigenvector, v, of B maximizes $x^T Bx$.

$Bv = \lambda v$ for maximum λ.
$\rightarrow vBv = \lambda$ for unit v.

Eigenvectors of $A^T A$ form orthonormal basis.

Intuition: $\max \frac{1}{2} v^T Bv + \lambda(1 - |v|^2)$
$\rightarrow Bv = \lambda v$.

Any other vector $av + (1-a)x$, $x \cdot v = 0$
x is composed of possibly smaller eigenvalue vectors.

$\rightarrow vBv \geq (av + (1-a)x)^T B(av + (1-a)x) = a^2 v^T Bv + (1-a)^2 x^T Bx$
for v, $av + (1-a)x$.

Computing eigenvalues.

Power method:

Computing eigenvalues.

Power method:
Choose random unit x.

Computing eigenvalues.

Power method:
Choose random unit x.
Repeat: Let $x = Bx$. Scale x to unit vector.

Computing eigenvalues.

Power method:

Choose random unit x.

Repeat: Let $x = Bx$. Scale x to unit vector.

Expected projection of random x onto unit v?

Computing eigenvalues.

Power method:
Choose random unit x.
Repeat: Let $x = Bx$. Scale x to unit vector.

Expected projection of random x onto unit v?
$\frac{1}{d}$!

Computing eigenvalues.

Power method:
Choose random unit x.
Repeat: Let $x = Bx$. Scale x to unit vector.

Expected projection of random x onto unit v?
$\frac{1}{d}$! First coordinate in random rotation!

Computing eigenvalues.

Power method:
Choose random unit x.
Repeat: Let $x = Bx$. Scale x to unit vector.

Expected projection of random x onto unit v?
$\frac{1}{d}$! First coordinate in random rotation!

$x = a_1 v_1 + a_2 v_2 + \cdots$

Computing eigenvalues.

Power method:
  Choose random unit x.
   Repeat: Let $x = Bx$. Scale x to unit vector.

Expected projection of random x onto unit v?
  $\frac{1}{d}$! First coordinate in random rotation!

$x = a_1 v_1 + a_2 v_2 + \cdots$

$x_t \propto B^t x = a_1 \lambda_1^t v_1 + a_2 \lambda_2^t v_2 + \cdots$

Computing eigenvalues.

Power method:
Choose random unit x.
Repeat: Let $x = Bx$. Scale x to unit vector.

Expected projection of random x onto unit v?
$\frac{1}{d}$! First coordinate in random rotation!

$x = a_1 v_1 + a_2 v_2 + \cdots$

$x_t \propto B^t x = a_1 \lambda_1^t v_1 + a_2 \lambda_2^t v_2 + \cdots$

Mostly v_1 after a while since $\lambda_1^t > \lambda_2^t$.

Computing eigenvalues.

Power method:

Choose random unit x.

Repeat: Let $x = Bx$. Scale x to unit vector.

Expected projection of random x onto unit v?

$\frac{1}{d}$! First coordinate in random rotation!

$x = a_1 v_1 + a_2 v_2 + \cdots$

$x_t \propto B^t x = a_1 \lambda_1^t v_1 + a_2 \lambda_2^t v_2 + \cdots$

Mostly v_1 after a while since $\lambda_1^t > \lambda_2^t$.

If no gap, then any vector in subspace of $[v_1, v_2]$ is fine.

Cluster Algorithm.

Power method:

Cluster Algorithm.

Power method:
Choose random unit x.

Cluster Algorithm.

Power method:

Choose random unit x.

Repeat: Let $x = Bx$. Scale x to unit vector.

Cluster Algorithm.

Power method:

Choose random unit x.

Repeat: Let $x = Bx$. Scale x to unit vector.

Expected projection of random x onto unit v?

Cluster Algorithm.

Power method:
Choose random unit x.
Repeat: Let $x = Bx$. Scale x to unit vector.

Expected projection of random x onto unit v?

Cluster Algorithm.

Power method:
Choose random unit x.
Repeat: Let $x = Bx$. Scale x to unit vector.

Expected projection of random x onto unit v?

2-means Algorithm:

Cluster Algorithm.

Power method:
Choose random unit x.
Repeat: Let $x = Bx$. Scale x to unit vector.

Expected projection of random x onto unit v?

2-means Algorithm:
Choose random partition.

Cluster Algorithm.

Power method:
Choose random unit x.
Repeat: Let $x = Bx$. Scale x to unit vector.

Expected projection of random x onto unit v?

2-means Algorithm:
Choose random partition.
Repeat: Compute means of partition. Project, cluster.

Cluster Algorithm.

Power method:
Choose random unit x.
Repeat: Let $x = Bx$. Scale x to unit vector.

Expected projection of random x onto unit v?

2-means Algorithm:
Choose random partition.
Repeat: Compute means of partition. Project, cluster.

Choose random $+1/-1$ vector.

Cluster Algorithm.

Power method:
Choose random unit x.
Repeat: Let $x = Bx$. Scale x to unit vector.

Expected projection of random x onto unit v?

2-means Algorithm:
Choose random partition.
Repeat: Compute means of partition. Project, cluster.

Choose random $+1/-1$ vector.
Repeat:

Cluster Algorithm.

Power method:
Choose random unit x.
Repeat: Let $x = Bx$. Scale x to unit vector.

Expected projection of random x onto unit v?

2-means Algorithm:
Choose random partition.
Repeat: Compute means of partition. Project, cluster.

Choose random $+1/-1$ vector.
Repeat:
$\times$ by A^T (direction between means)!

Cluster Algorithm.

Power method:
Choose random unit x.
Repeat: Let $x = Bx$. Scale x to unit vector.

Expected projection of random x onto unit v?

2-means Algorithm:
Choose random partition.
Repeat: Compute means of partition. Project, cluster.

Choose random $+1/-1$ vector.
Repeat:
$\times$ by A^T (direction between means)!
$\times$ by A (project points on direction),

Cluster Algorithm.

Power method:
Choose random unit x.
Repeat: Let $x = Bx$. Scale x to unit vector.

Expected projection of random x onto unit v?

2-means Algorithm:
Choose random partition.
Repeat: Compute means of partition. Project, cluster.

Choose random $+1/-1$ vector.
Repeat:
$\times$ by A^T (direction between means)!
$\times$ by A (project points on direction),
Cluster (round to +1/-1 vector.)

$$A^T \tilde{x} = \begin{bmatrix} x_1^T & x_2^T & \cdots \end{bmatrix} \times \begin{bmatrix} +1 \\ -1 \\ \cdots \\ +1 \end{bmatrix}$$

$$Av = \begin{bmatrix} \cdots x_1 \cdots \\ \cdots x_2 \cdots \\ \cdots \\ x_n \end{bmatrix} \times \begin{bmatrix} \vdots \\ m_{+1} - m_{-1} \\ \vdots \end{bmatrix}$$

Cluster Algorithm.

Power method:
Choose random unit x.
Repeat: Let $x = Bx$. Scale x to unit vector.

Expected projection of random x onto unit v?

2-means Algorithm:
Choose random partition.
Repeat: Compute means of partition. Project, cluster.

Choose random $+1/-1$ vector.
Repeat:
$\times$ by A^T (direction between means)!
$\times$ by A (project points on direction),
Cluster (round to +1/-1 vector.)

$$A^T \tilde{x} = \begin{bmatrix} x_1^T & x_2^T & \cdots \end{bmatrix} \times \begin{bmatrix} +1 \\ -1 \\ \cdots \\ +1 \end{bmatrix}$$

$$Av = \begin{bmatrix} \cdots x_1 \cdots \\ \cdots x_2 \cdots \\ \cdots \\ x_n \end{bmatrix} \times \begin{bmatrix} \vdots \\ m_{+1} - m_{-1} \\ \vdots \end{bmatrix}$$

Repeatedly multiplying by AA^T, with rounding step.

Cluster Algorithm.

Power method:
Choose random unit x.
Repeat: Let $x = Bx$. Scale x to unit vector.

Expected projection of random x onto unit v?

2-means Algorithm:
Choose random partition.
Repeat: Compute means of partition. Project, cluster.

Choose random $+1/-1$ vector.
Repeat:
$\times$ by A^T (direction between means)!
$\times$ by A (project points on direction),
Cluster (round to +1/-1 vector.)

$$A^T\tilde{x} = \begin{bmatrix} x_1^T & x_2^T & \cdots \end{bmatrix} \times \begin{bmatrix} +1 \\ -1 \\ \cdots \\ +1 \end{bmatrix}$$

$$Av = \begin{bmatrix} \cdots x_1 \cdots \\ \cdots x_2 \cdots \\ \cdots \\ x_n \end{bmatrix} \times \begin{bmatrix} \vdots \\ m_{+1} - m_{-1} \\ \vdots \end{bmatrix}$$

Repeatedly multiplying by AA^T, with rounding step.
Power method with rounding.

Cluster Like algorithm

Given random graph on V_1 and V_2, and $p = q + \delta$.

Cluster Like algorithm

Given random graph on V_1 and V_2, and $p = q + \delta$.

$Pr[edge(x,y) : x, y \in V_i] = p \quad Pr[edge(x,y) : x \in V_1, y \in V_2] = q$

Cluster Like algorithm

Given random graph on V_1 and V_2, and $p = q + \delta$.

$Pr[edge(x,y) : x,y \in V_i] = p \quad Pr[edge(x,y) : x \in V_1, y \in V_2] = q$

Algorithm:

Cluster Like algorithm

Given random graph on V_1 and V_2, and $p = q + \delta$.

$Pr[edge(x,y) : x, y \in V_i] = p \quad Pr[edge(x,y) : x \in V_1, y \in V_2] = q$

Algorithm:
Randomly split vertices into two groups. (± 1)
Repeat: Switch if majority of neighbors on other side.

Cluster Like algorithm

Given random graph on V_1 and V_2, and $p = q + \delta$.

$Pr[edge(x,y) : x, y \in V_i] = p \quad Pr[edge(x,y) : x \in V_1, y \in V_2] = q$

Algorithm:
Randomly split vertices into two groups. (± 1)
Repeat: Switch if majority of neighbors on other side.

Cluster Like algorithm

Given random graph on V_1 and V_2, and $p = q + \delta$.

$Pr[edge(x,y) : x, y \in V_i] = p \quad Pr[edge(x,y) : x \in V_1, y \in V_2] = q$

Algorithm:
Randomly split vertices into two groups. (± 1)
Repeat: Switch if majority of neighbors on other side.

Analyz-easy Algorithm.

Recall:

Analyz-easy Algorithm.

Recall: Random vector has projection $1/\sqrt{d}$ on first eigenvector.
$p = q + \delta$. p is probability of same side guy.

Analyz-easy Algorithm.

Recall: Random vector has projection $1/\sqrt{d}$ on first eigenvector.
$p = q + \delta$. p is probability of same side guy.

Split vertices into $S_1, \ldots, S_k$.

Analyz-easy Algorithm.

Recall: Random vector has projection $1/\sqrt{d}$ on first eigenvector.
$p = q + \delta$. p is probability of same side guy.

Split vertices into $S_1, \ldots, S_k$.

Randomly split S_1 into two groups. (± 1)
Repeat for $i > 1$: Place each vertex of S_i majority side of neighbors in S_{i-1}.

Analyz-easy Algorithm.

Recall: Random vector has projection $1/\sqrt{d}$ on first eigenvector.
$p = q + \delta$. p is probability of same side guy.

Split vertices into $S_1, \ldots, S_k$.

Randomly split S_1 into two groups. (± 1)
Repeat for $i > 1$: Place each vertex of S_i majority side of neighbors in S_{i-1}.

Analyz-easy Algorithm.

Recall: Random vector has projection $1/\sqrt{d}$ on first eigenvector.
$p = q + \delta$. p is probability of same side guy.

Split vertices into $S_1, \ldots, S_k$.

Randomly split S_1 into two groups. (± 1)
Repeat for $i > 1$: Place each vertex of S_i majority side of neighbors in S_{i-1}.

At first: partition has $1/2 + 1/\sqrt{n}$ imbalance.

Analyz-easy Algorithm.

Recall: Random vector has projection $1/\sqrt{d}$ on first eigenvector.
$p = q + \delta$. p is probability of same side guy.

Split vertices into $S_1, \ldots, S_k$.

Randomly split S_1 into two groups. (± 1)
Repeat for $i > 1$: Place each vertex of S_i majority side of neighbors in S_{i-1}.

At first: partition has $1/2 + 1/\sqrt{n}$ imbalance.
S_1^1 is more V_1, S_1^2 is more V_2.

Analyz-easy Algorithm.

Recall: Random vector has projection $1/\sqrt{d}$ on first eigenvector.
$p = q + \delta$. p is probability of same side guy.

Split vertices into $S_1, \ldots, S_k$.

Randomly split S_1 into two groups. (± 1)
Repeat for $i > 1$: Place each vertex of S_i majority side of neighbors in S_{i-1}.

At first: partition has $1/2 + 1/\sqrt{n}$ imbalance.
S_1^1 is more V_1, S_1^2 is more V_2.
Let $p_t = Pr[v \in S_1^1 \text{in iteration } t | v \in V_1]$
$p_0 = 1/2 + 1/\sqrt{n}$.

Analyz-easy Algorithm.

Recall: Random vector has projection $1/\sqrt{d}$ on first eigenvector.
$p = q + \delta$. p is probability of same side guy.

Split vertices into $S_1, \ldots, S_k$.

Randomly split S_1 into two groups. (± 1)
Repeat for $i > 1$: Place each vertex of S_i majority side of neighbors in S_{i-1}.

At first: partition has $1/2 + 1/\sqrt{n}$ imbalance.
S_1^1 is more V_1, S_1^2 is more V_2.
Let $p_t = Pr[v \in S_1^1 \text{in iteration } t | v \in V_1]$
$p_0 = 1/2 + 1/\sqrt{n}$.

Random edge advantage: $(p-q)\frac{|S_1|}{\sqrt{n}}$ out of $|S_1|$:

Analyz-easy Algorithm.

Recall: Random vector has projection $1/\sqrt{d}$ on first eigenvector.
$p = q + \delta$. p is probability of same side guy.

Split vertices into $S_1, \ldots, S_k$.

Randomly split S_1 into two groups. (± 1)
Repeat for $i > 1$: Place each vertex of S_i majority side of neighbors in S_{i-1}.

At first: partition has $1/2 + 1/\sqrt{n}$ imbalance.
S_1^1 is more V_1, S_1^2 is more V_2.
Let $p_t = Pr[v \in S_1^1 \text{in iteration } t | v \in V_1]$
$p_0 = 1/2 + 1/\sqrt{n}$.

Random edge advantage: $(p-q)\frac{|S_1|}{\sqrt{n}}$ out of $|S_1|$: $\Delta = \frac{p-q}{\sqrt{n}}$.

Analyz-easy Algorithm.

Recall: Random vector has projection $1/\sqrt{d}$ on first eigenvector.
$p = q + \delta$. p is probability of same side guy.

Split vertices into $S_1, \ldots, S_k$.

Randomly split S_1 into two groups. (± 1)
Repeat for $i > 1$: Place each vertex of S_i majority side of neighbors in S_{i-1}.

At first: partition has $1/2 + 1/\sqrt{n}$ imbalance.
S_1^1 is more V_1, S_1^2 is more V_2.
Let $p_t = Pr[v \in S_1^1$ in iteration $t | v \in V_1]$
$p_0 = 1/2 + 1/\sqrt{n}$.

Random edge advantage: $(p-q)\frac{|S_1|}{\sqrt{n}}$ out of $|S_1|$: $\Delta = \frac{p-q}{\sqrt{n}}$.

Samples: $d = \frac{(p+q)|S_1|}{2}$.

Analyz-easy Algorithm.

Recall: Random vector has projection $1/\sqrt{d}$ on first eigenvector.
$p = q + \delta$. p is probability of same side guy.

Split vertices into $S_1, \ldots, S_k$.

Randomly split S_1 into two groups. (± 1)
Repeat for $i > 1$: Place each vertex of S_i majority side of neighbors in S_{i-1}.

At first: partition has $1/2 + 1/\sqrt{n}$ imbalance.
S_1^1 is more V_1, S_1^2 is more V_2.
Let $p_t = Pr[v \in S_1^1 \text{in iteration } t | v \in V_1]$
$p_0 = 1/2 + 1/\sqrt{n}$.

Random edge advantage: $(p-q)\frac{|S_1|}{\sqrt{n}}$ out of $|S_1|$: $\Delta = \frac{p-q}{\sqrt{n}}$.

Samples: $d = \frac{(p+q)|S_1|}{2}$.
$Pr[B(1/2 + \Delta, d) \geq d/2] \geq \frac{1}{2} + c\frac{\Delta}{\sqrt{d}}$.

Analyz-easy Algorithm.

Recall: Random vector has projection $1/\sqrt{d}$ on first eigenvector.
$p = q + \delta$. p is probability of same side guy.

Split vertices into $S_1, \ldots, S_k$.

Randomly split S_1 into two groups. (± 1)
Repeat for $i > 1$: Place each vertex of S_i majority side of neighbors in S_{i-1}.

At first: partition has $1/2 + 1/\sqrt{n}$ imbalance.
S_1^1 is more V_1, S_1^2 is more V_2.
Let $p_t = Pr[v \in S_1^1 \text{in iteration } t | v \in V_1]$
$p_0 = 1/2 + 1/\sqrt{n}$.

Random edge advantage: $(p-q)\frac{|S_1|}{\sqrt{n}}$ out of $|S_1|$: $\Delta = \frac{p-q}{\sqrt{n}}$.

Samples: $d = \frac{(p+q)|S_1|}{2}$.
$Pr[B(1/2+\Delta, d) \geq d/2] \geq \frac{1}{2} + c\frac{\Delta}{\sqrt{d}}$.
Gaussian Picture.

Analyz-easy Algorithm.

Recall: Random vector has projection $1/\sqrt{d}$ on first eigenvector.
$p = q + \delta$. p is probability of same side guy.

Split vertices into $S_1, \ldots, S_k$.

Randomly split S_1 into two groups. (± 1)
Repeat for $i > 1$: Place each vertex of S_i majority side of neighbors in S_{i-1}.

At first: partition has $1/2 + 1/\sqrt{n}$ imbalance.
S_1^1 is more V_1, S_1^2 is more V_2.
Let $p_t = Pr[v \in S_1^1$ in iteration $t | v \in V_1]$
$p_0 = 1/2 + 1/\sqrt{n}$.

Random edge advantage: $(p-q)\frac{|S_1|}{\sqrt{n}}$ out of $|S_1|$: $\Delta = \frac{p-q}{\sqrt{n}}$.

Samples: $d = \frac{(p+q)|S_1|}{2}$.
$Pr[B(1/2 + \Delta, d) \geq d/2] \geq \frac{1}{2} + c\frac{\Delta}{\sqrt{d}}$.
Gaussian Picture. Berry-Esseen: converges in middle!

Analyz-easy Algorithm.

Recall: Random vector has projection $1/\sqrt{d}$ on first eigenvector.
$p = q + \delta$. p is probability of same side guy.

Split vertices into $S_1, \ldots, S_k$.

Randomly split S_1 into two groups. (± 1)
Repeat for $i > 1$: Place each vertex of S_i majority side of neighbors in S_{i-1}.

At first: partition has $1/2 + 1/\sqrt{n}$ imbalance.
S_1^1 is more V_1, S_1^2 is more V_2.
Let $p_t = Pr[v \in S_1^1 \text{in iteration } t | v \in V_1]$
$p_0 = 1/2 + 1/\sqrt{n}$.

Random edge advantage: $(p-q)\frac{|S_1|}{\sqrt{n}}$ out of $|S_1|$: $\Delta = \frac{p-q}{\sqrt{n}}$.

Samples: $d = \frac{(p+q)|S_1|}{2}$.
$Pr[B(1/2+\Delta, d) \geq d/2] \geq \frac{1}{2} + c\frac{\Delta}{\sqrt{d}}$.
Gaussian Picture. Berry-Esseen: converges in middle!

If $\frac{(p-q)d}{\sqrt{d}\sqrt{n}} \geq \frac{2}{\sqrt{n}}$.

Analyz-easy Algorithm.

Recall: Random vector has projection $1/\sqrt{d}$ on first eigenvector.
$p = q + \delta$. p is probability of same side guy.

Split vertices into $S_1, \ldots, S_k$.

Randomly split S_1 into two groups. (± 1)
Repeat for $i > 1$: Place each vertex of S_i majority side of neighbors in S_{i-1}.

At first: partition has $1/2 + 1/\sqrt{n}$ imbalance.
S_1^1 is more V_1, S_1^2 is more V_2.
Let $p_t = Pr[v \in S_1^1 \text{in iteration } t | v \in V_1]$
$p_0 = 1/2 + 1/\sqrt{n}$.

Random edge advantage: $(p-q)\frac{|S_1|}{\sqrt{n}}$ out of $|S_1|$: $\Delta = \frac{p-q}{\sqrt{n}}$.

Samples: $d = \frac{(p+q)|S_1|}{2}$.
$Pr[B(1/2 + \Delta, d) \geq d/2] \geq \frac{1}{2} + c\frac{\Delta}{\sqrt{d}}$.
Gaussian Picture. Berry-Esseen: converges in middle!

If $\frac{(p-q)d}{\sqrt{d}\sqrt{n}} \geq \frac{2}{\sqrt{n}}$. $\implies$

Analyz-easy Algorithm.

Recall: Random vector has projection $1/\sqrt{d}$ on first eigenvector.
$p = q + \delta$. p is probability of same side guy.

Split vertices into $S_1, \ldots, S_k$.

Randomly split S_1 into two groups. (± 1)
Repeat for $i > 1$: Place each vertex of S_i majority side of neighbors in S_{i-1}.

At first: partition has $1/2 + 1/\sqrt{n}$ imbalance.
S_1^1 is more V_1, S_1^2 is more V_2.
Let $p_t = Pr[v \in S_1^1 \text{in iteration } t | v \in V_1]$
$p_0 = 1/2 + 1/\sqrt{n}$.

Random edge advantage: $(p-q)\frac{|S_1|}{\sqrt{n}}$ out of $|S_1|$: $\Delta = \frac{p-q}{\sqrt{n}}$.

Samples: $d = \frac{(p+q)|S_1|}{2}$.
$Pr[B(1/2+\Delta, d) \geq d/2] \geq \frac{1}{2} + c\frac{\Delta}{\sqrt{d}}$.
Gaussian Picture. Berry-Esseen: converges in middle!

If $\frac{(p-q)d}{\sqrt{d}\sqrt{n}} \geq \frac{2}{\sqrt{n}}$. $\implies \frac{(p-q)d}{\sqrt{d}} = \delta\sqrt{d} \geq 2 \implies$ Yaay!

Analyz-easy Algorithm.

Recall: Random vector has projection $1/\sqrt{d}$ on first eigenvector.
$p = q + \delta$. p is probability of same side guy.

Split vertices into $S_1, \ldots, S_k$.

Randomly split S_1 into two groups. (± 1)
Repeat for $i > 1$: Place each vertex of S_i majority side of neighbors in S_{i-1}.

At first: partition has $1/2 + 1/\sqrt{n}$ imbalance.
S_1^1 is more V_1, S_1^2 is more V_2.
Let $p_t = Pr[v \in S_1^1 \text{in iteration } t | v \in V_1]$
$p_0 = 1/2 + 1/\sqrt{n}$.

Random edge advantage: $(p-q)\frac{|S_1|}{\sqrt{n}}$ out of $|S_1|$: $\Delta = \frac{p-q}{\sqrt{n}}$.

Samples: $d = \frac{(p+q)|S_1|}{2}$.
$Pr[B(1/2+\Delta, d) \geq d/2] \geq \frac{1}{2} + c\frac{\Delta}{\sqrt{d}}$.
Gaussian Picture. Berry-Esseen: converges in middle!

If $\frac{(p-q)d}{\sqrt{d}\sqrt{n}} \geq \frac{2}{\sqrt{n}}$. $\implies$ $\frac{(p-q)d}{\sqrt{d}} = \delta\sqrt{d} \geq 2 \implies$ Yaay!

Sizes of S_i? d is $\propto S_i$, advantage grows, $|S_1| = \Theta(n)$, others smaller.

Gaussian Shift advantage.

Let $\Delta = (p - q) \times$ imbalance.

Gaussian Shift advantage.

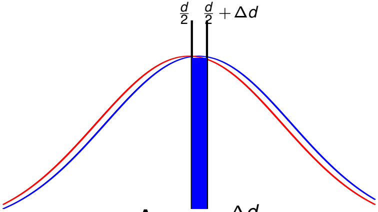
Let $\Delta = (p - q) \times$ imbalance.

Step 1: $\Delta = (p - q) \times \frac{1}{\sqrt{n}}$.

Gaussian Shift advantage.

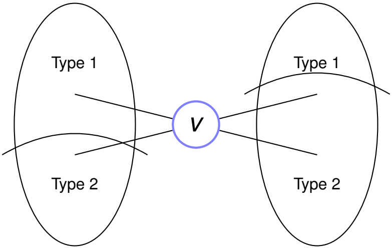
Let $\Delta = (p - q) \times$ imbalance.

Step 1: $\Delta = (p - q) \times \frac{1}{\sqrt{n}}$.



Area $\approx \frac{\Delta d}{\sqrt{d}}$

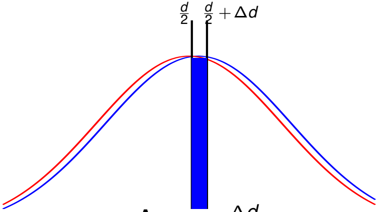
Assuming $\Delta d << \sqrt{d}$.



Gaussian Shift advantage.

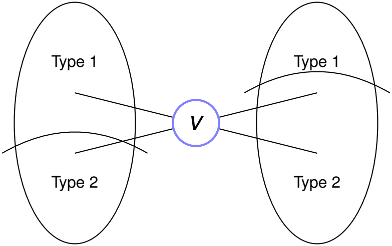
Let $\Delta = (p-q) \times$ imbalance.

Step 1: $\Delta = (p-q) \times \frac{1}{\sqrt{n}}$.



Area $\approx \frac{\Delta d}{\sqrt{d}}$

Assuming $\Delta d << \sqrt{d}$.

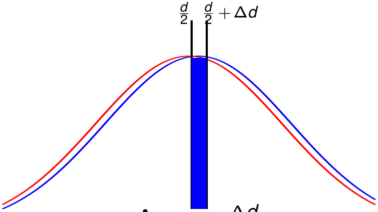


Original imbalance is $1/\sqrt{n}$ of the vertices.

Gaussian Shift advantage.

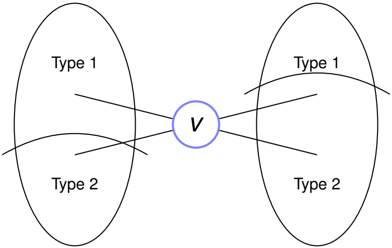
Let $\Delta = (p-q) \times$ imbalance.

Step 1: $\Delta = (p-q) \times \frac{1}{\sqrt{n}}$.



Area $\approx \frac{\Delta d}{\sqrt{d}}$

Assuming $\Delta d << \sqrt{d}$.



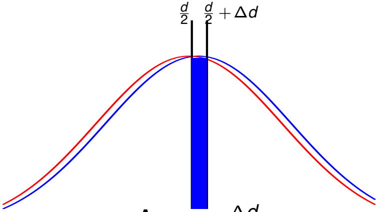
Original imbalance is $1/\sqrt{n}$ of the vertices.

If advantage $((p-q)/\sqrt{n})/\sqrt{d} >> 1/\sqrt{n}$, then imbalance increases.

Gaussian Shift advantage.

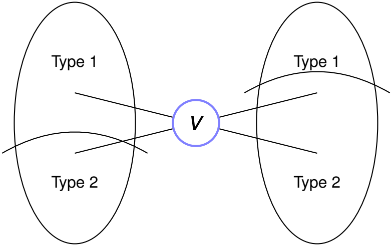
Let $\Delta = (p - q) \times$ imbalance.

Step 1: $\Delta = (p - q) \times \frac{1}{\sqrt{n}}$.



Area $\approx \frac{\Delta d}{\sqrt{d}}$

Assuming $\Delta d << \sqrt{d}$.



Original imbalance is $1/\sqrt{n}$ of the vertices.

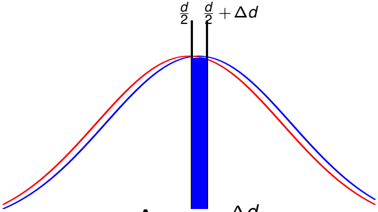
If advantage $((p - q)/\sqrt{n})/\sqrt{d} >> 1/\sqrt{n}$, then imbalance increases.

...and then gets better after that.

Gaussian Shift advantage.

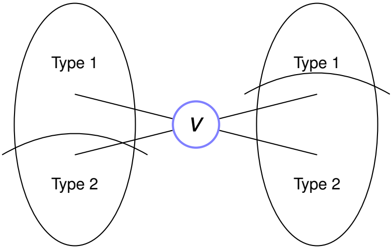
Let $\Delta = (p-q)\times$ imbalance.

Step 1: $\Delta = (p-q) \times \frac{1}{\sqrt{n}}$.



Area $\approx \frac{\Delta d}{\sqrt{d}}$

Assuming $\Delta d << \sqrt{d}$.



Original imbalance is $1/\sqrt{n}$ of the vertices.

If advantage $((p-q)/\sqrt{n})/\sqrt{d} >> 1/\sqrt{n}$, then imbalance increases.

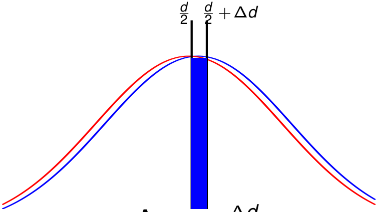
...and then gets better after that.

Roughly: $(p-q) \geq \sqrt{d}$.

Gaussian Shift advantage.

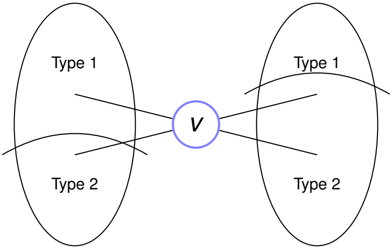
Let $\Delta = (p - q) \times$ imbalance.

Step 1: $\Delta = (p - q) \times \frac{1}{\sqrt{n}}$.



Area $\approx \frac{\Delta d}{\sqrt{d}}$

Assuming $\Delta d << \sqrt{d}$.



Original imbalance is $1/\sqrt{n}$ of the vertices.

If advantage $((p - q)/\sqrt{n})/\sqrt{d} >> 1/\sqrt{n}$, then imbalance increases.

...and then gets better after that.

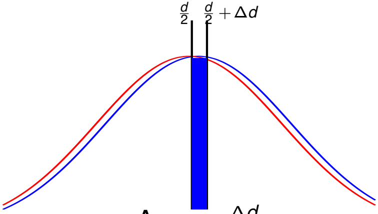
Roughly: $(p - q) \geq \sqrt{d}$.

Note: need $d >> \Omega(\log n)$.

Gaussian Shift advantage.

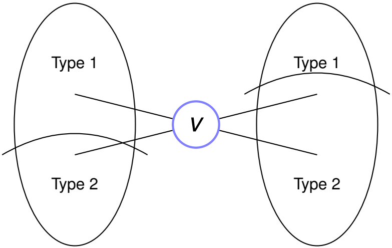
Let $\Delta = (p - q) \times$ imbalance.

Step 1: $\Delta = (p - q) \times \frac{1}{\sqrt{n}}$.



Area $\approx \frac{\Delta d}{\sqrt{d}}$

Assuming $\Delta d << \sqrt{d}$.



Original imbalance is $1/\sqrt{n}$ of the vertices.

If advantage $((p - q)/\sqrt{n})/\sqrt{d} >> 1/\sqrt{n}$, then imbalance increases.

...and then gets better after that.

Roughly: $(p - q) \geq \sqrt{d}$.

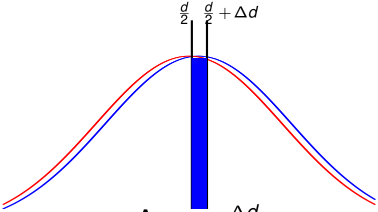
Note: need $d >> \Omega(\log n)$.

Otherwise there are disconnected vertices.

Gaussian Shift advantage.

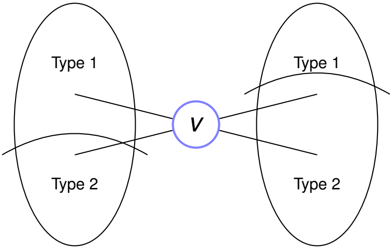
Let $\Delta = (p-q) \times$ imbalance.

Step 1: $\Delta = (p-q) \times \frac{1}{\sqrt{n}}$.



Area $\approx \frac{\Delta d}{\sqrt{d}}$

Assuming $\Delta d << \sqrt{d}$.



Original imbalance is $1/\sqrt{n}$ of the vertices.

If advantage $((p-q)/\sqrt{n})/\sqrt{d} >> 1/\sqrt{n}$, then imbalance increases.

...and then gets better after that.

Roughly: $(p-q) \geq \sqrt{d}$.

Note: need $d >> \Omega(\log n)$.
Otherwise there are disconnected vertices.

Sum up.

Clustering mixture of gaussians.

Sum up.

Clustering mixture of gaussians.

Near Neighbor works with sufficient data.

Sum up.

Clustering mixture of gaussians.

Near Neighbor works with sufficient data.

Projection onto subspace of means is better.

Sum up.

Clustering mixture of gaussians.

Near Neighbor works with sufficient data.

Projection onto subspace of means is better.

Principal compent analysis can find subspace of means.

Sum up.

Clustering mixture of gaussians.

Near Neighbor works with sufficient data.

Projection onto subspace of means is better.

Principal compent analysis can find subspace of means.

Power method computes principal component.

Sum up.

Clustering mixture of gaussians.

Near Neighbor works with sufficient data.

Projection onto subspace of means is better.

Principal compent analysis can find subspace of means.

Power method computes principal component.

Generic clustering algorithm is rounded version of power method.

See you on Tuesday.