

Tree Method

Junwei Lu



HARVARD
T.H. CHAN

SCHOOL OF PUBLIC HEALTH
Department of Biostatistics



Classification Review

- **Discriminative classifier:** Model $\mathbb{P}(Y|X)$

Logistic regression

- **Generative classifier:** Model $\mathbb{P}(X|Y)$

LDA, QDA, Naive Bayes

- **Model-free classifier**

SVM, KNN



Question: How to choose the method in practice?

Comparison of Classification Methods

n = number of samples, d = number of features

If $d \gg n$

Use logistic, linear SVM, LDA

Logistic depends on all training data, but SVM only depends on support vectors

If d is small, but n is intermediate

Use SVM with Gaussian kernels, QDA

If d is small, but n is large

Create more features, use logistic, linear SVM, LDA

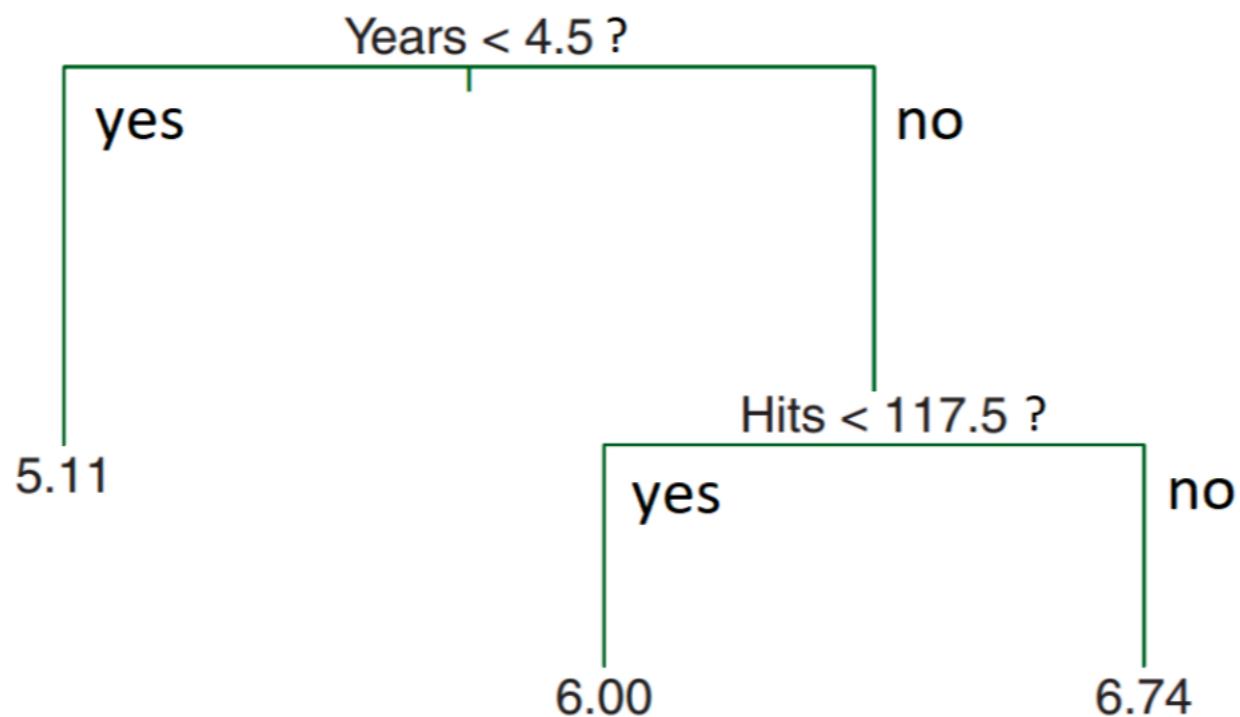
k-NN with no interoperability

In practice: Evaluate the performance on test data

Decision Tree

Decision tree is an **interoperable** method

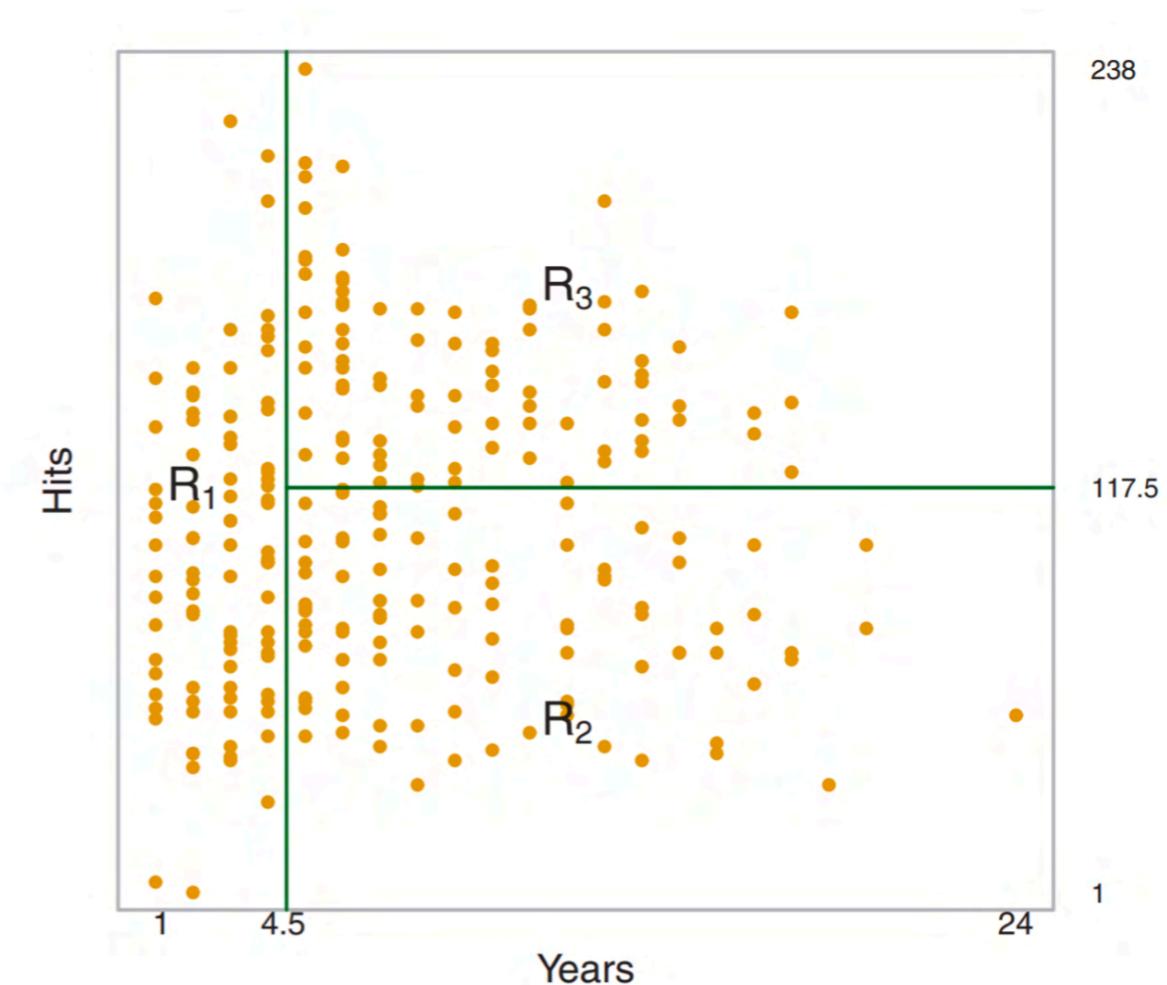
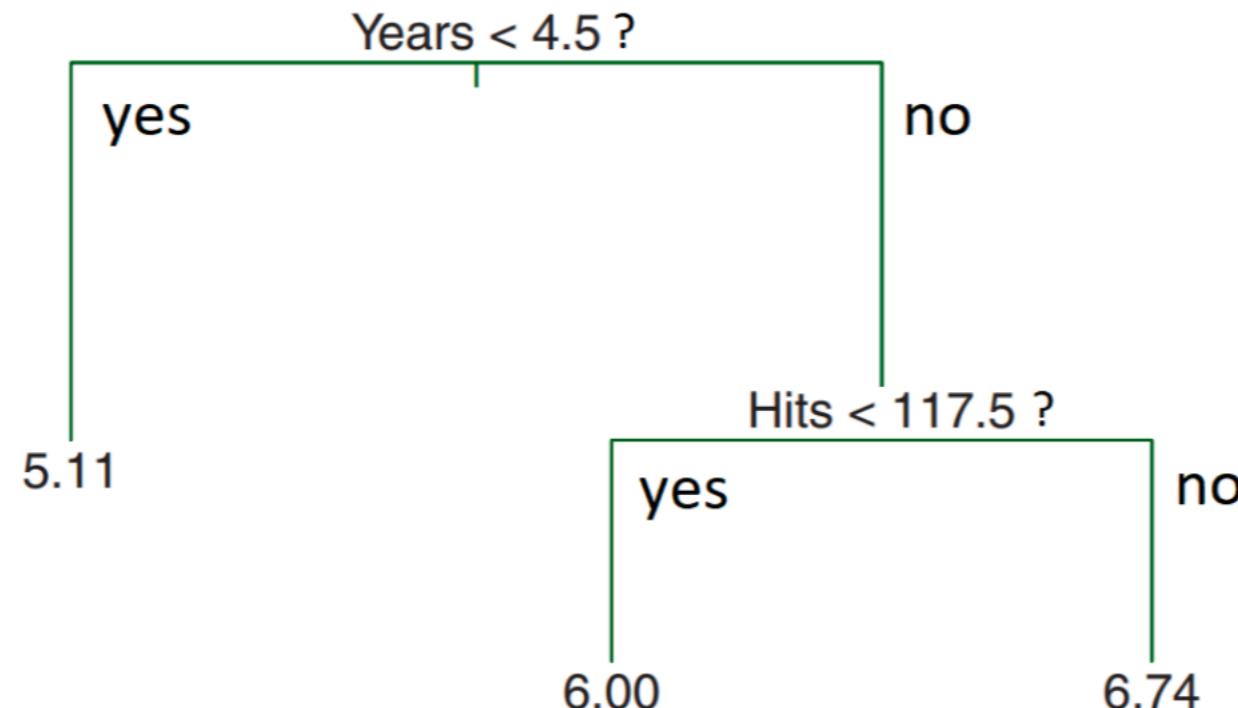
Example: Decide baseball play salaries



Decision Tree

Decision tree is an **interoperable** method

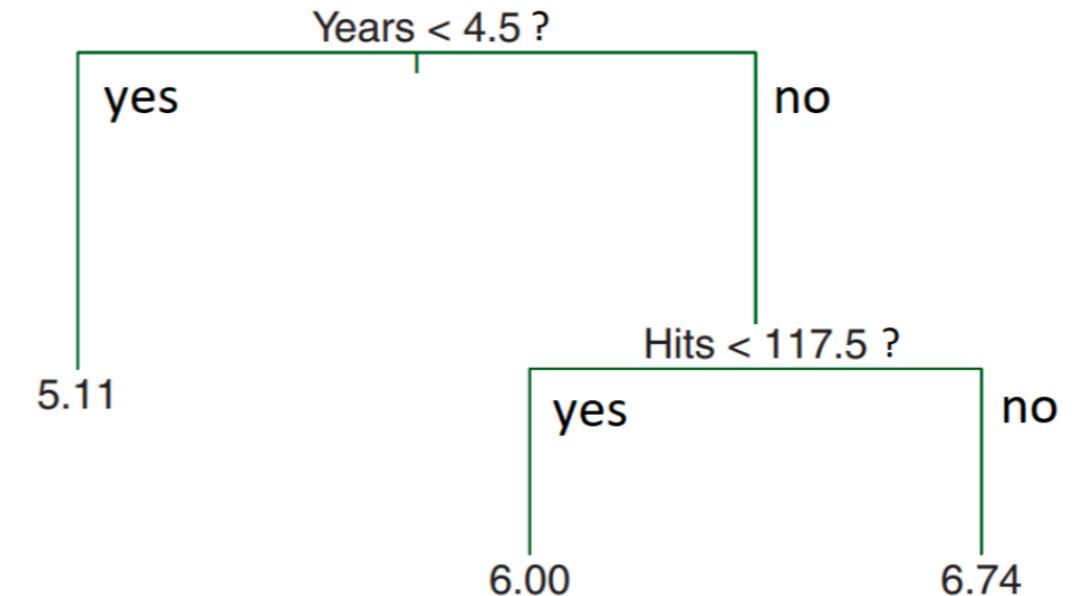
Example: Decide baseball play salaries



Concepts: Tree, leaf, node, child

Decision Tree

Tree leaf nodes \longleftrightarrow Regions

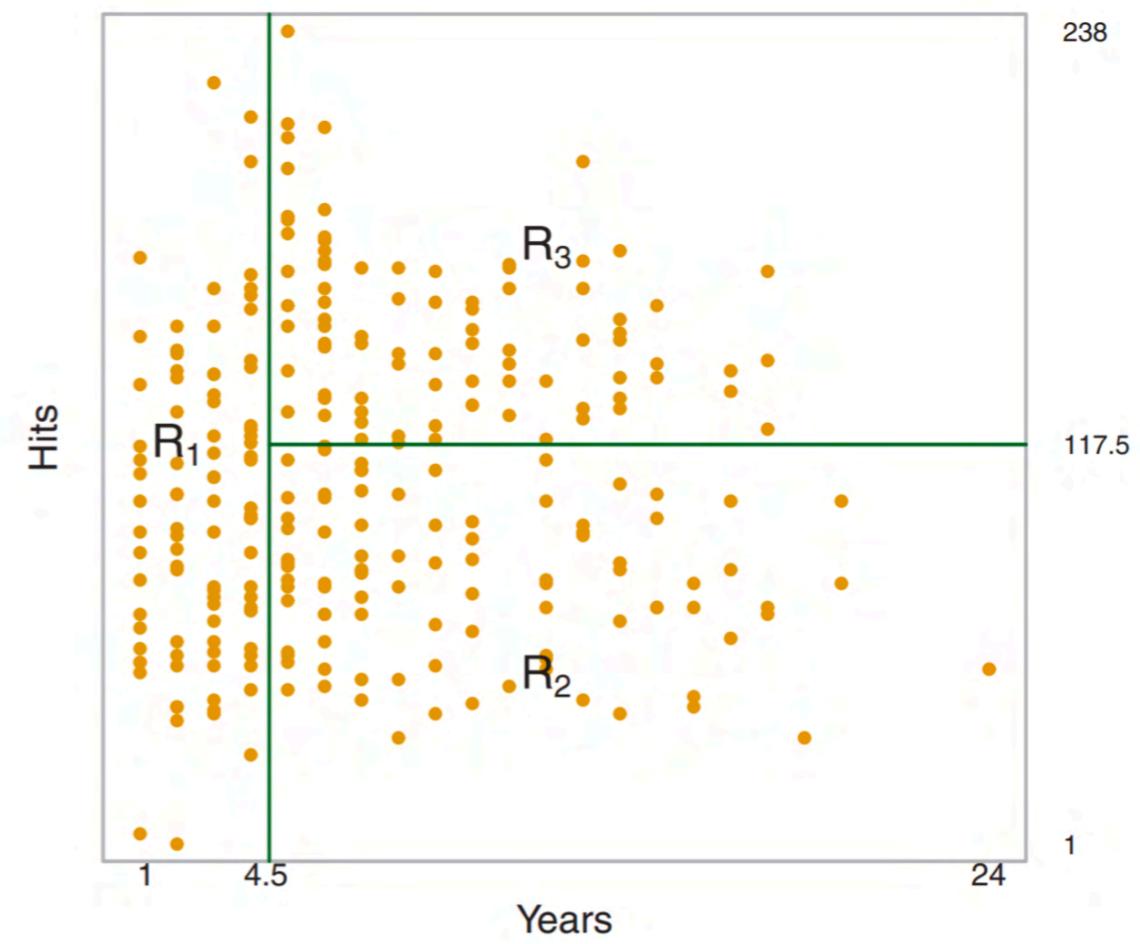


For test point x , if $x \in R_m$, predict $\hat{y} = \bar{y}_{R_m}$

$$\bar{y}_{R_m} = \frac{1}{|R_m|} \sum_{i:x_i \in R_m} y_i$$

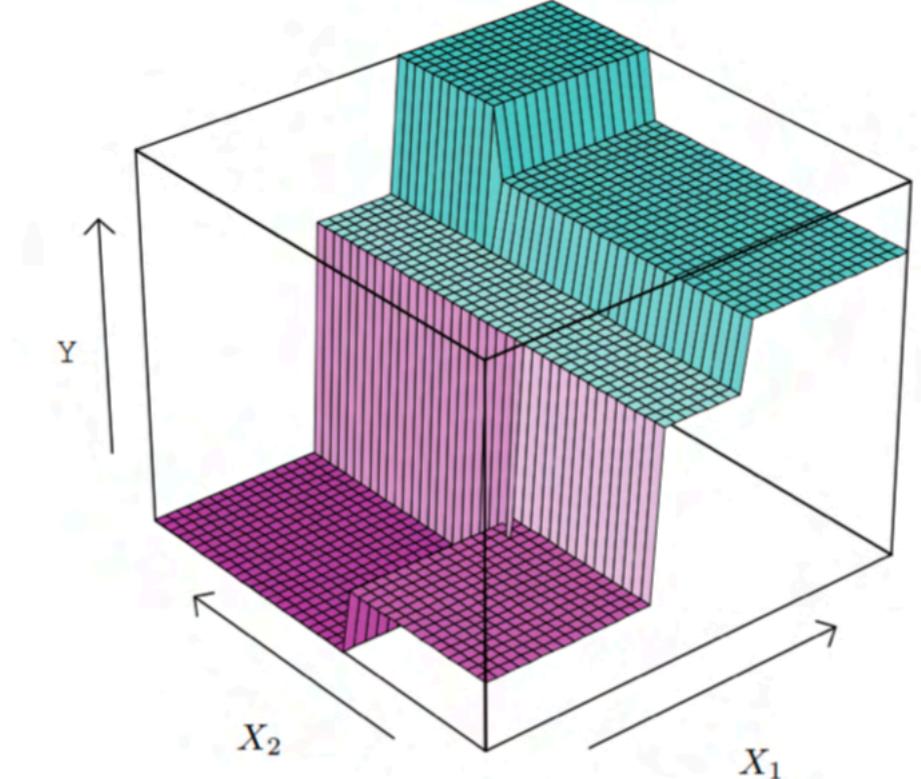
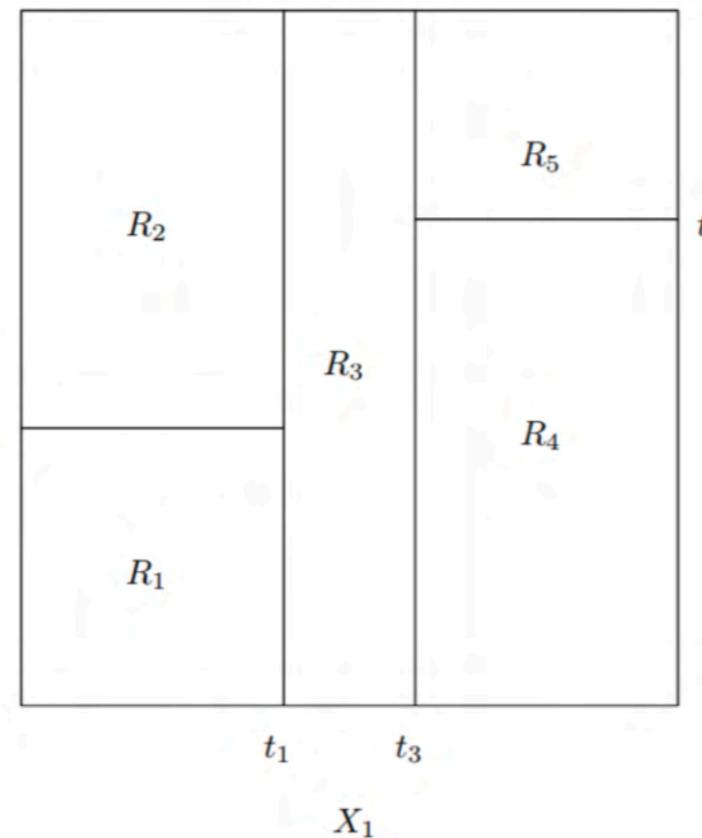
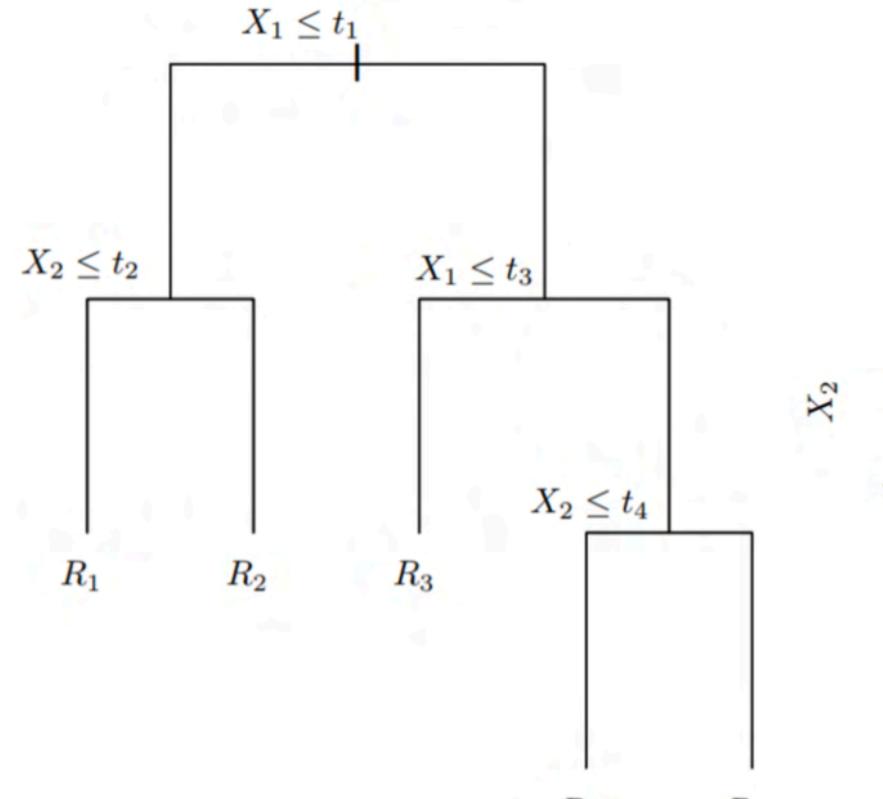
Predict the average of the outcome in the region

$$\hat{f}(x) = \sum_{m=1}^M \bar{y}_{R_m} \mathbb{I}(x \in R_m)$$

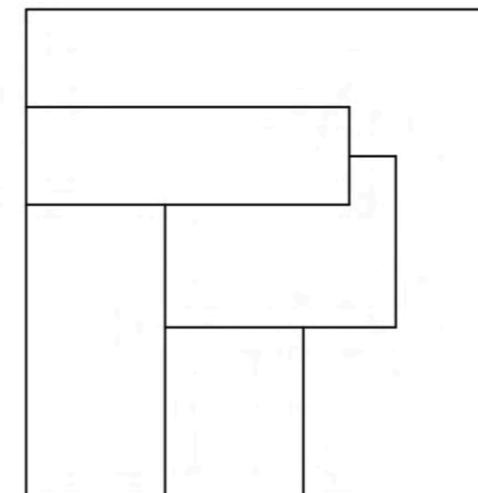
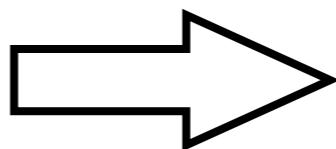


Decision Tree

The fitted function is piecewise constant $\mathcal{F} = \left\{ f(\mathbf{x}) = \sum_{m=1}^M \bar{y}_{R_m} \mathbb{I}(\mathbf{x} \in R_m) \right\}$



This image is impossible



Tree-based Regression Method



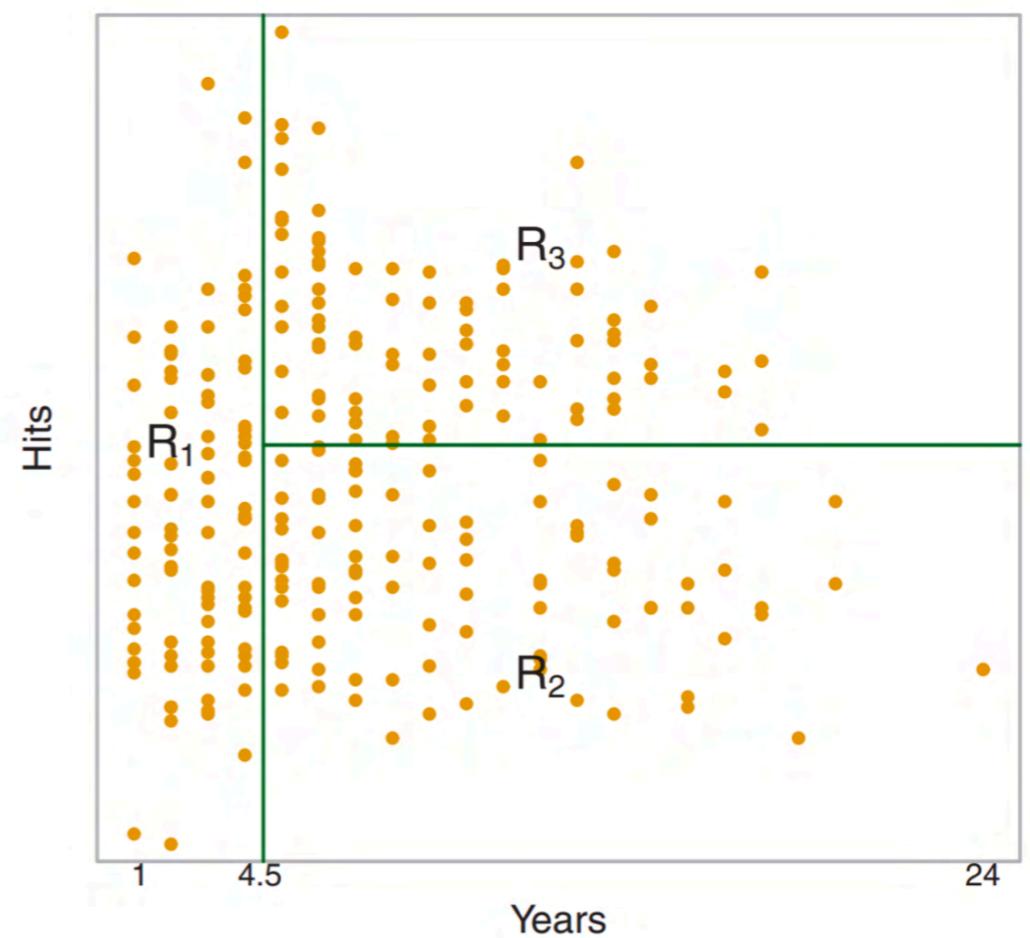
Question: How do we learn the tree from training data?

Step 1. Determine a loss function, e.g., ℓ_2 -loss: $R(f) = \mathbb{E}[Y - f(X)]^2$

$$\hat{R}(f) = \frac{1}{n} \sum_{i=1}^n (Y_i - f(X_i))^2$$

Since $f(\mathbf{x}) = \sum_{m=1}^M \bar{y}_{R_m} \mathbb{I}(\mathbf{x} \in R_m)$

$$\hat{R}(f) = \frac{1}{n} \sum_{m=1}^M \sum_{i:x_i \in R_m} (Y_i - \bar{y}_{R_m})^2$$



Classification and Regression Tree (CART)



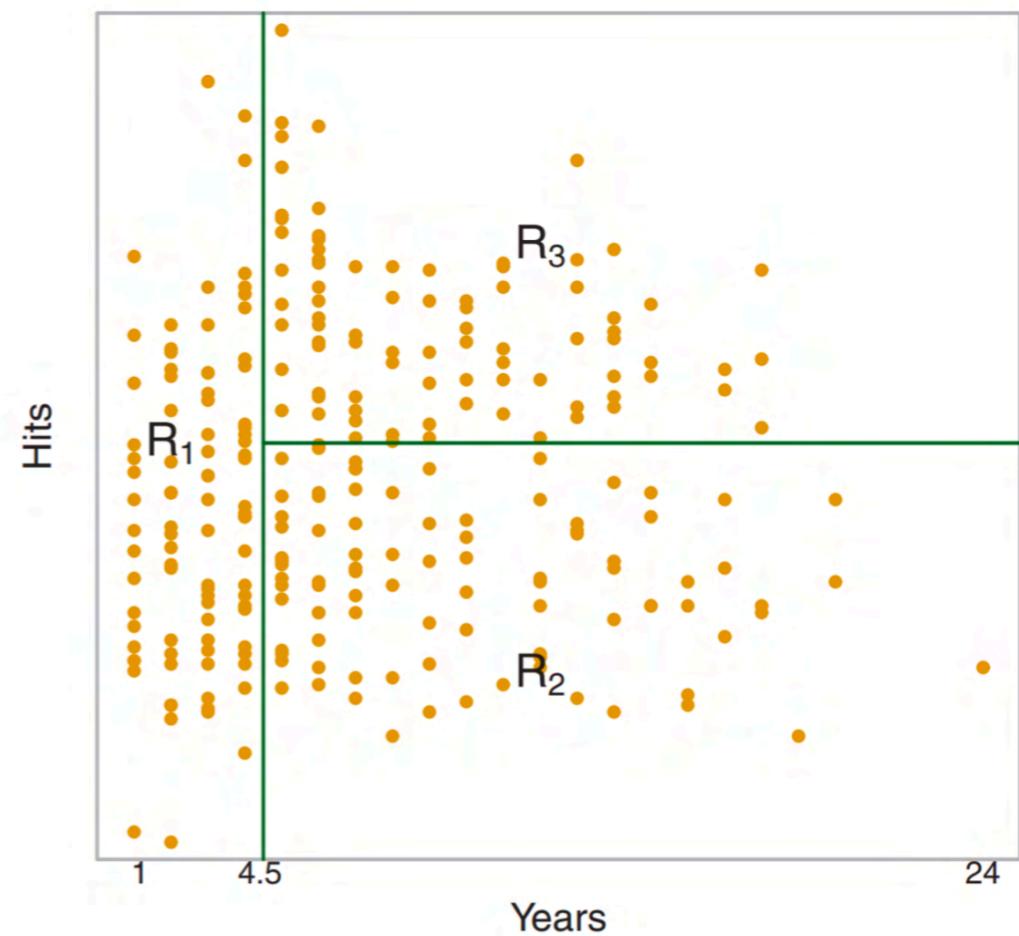
Question: How do we learn the tree from training data?

Step 1. Determine a loss function, e.g., ℓ_2 -loss: $R(f) = \mathbb{E}[Y - f(X)]^2$

$$\hat{R}(f) = \frac{1}{n} \sum_{m=1}^M \sum_{i:x_i \in R_m} (Y_i - \bar{y}_{R_m})^2$$

Step 2. Find the good “tree partition”

- Intractable to find the best tree
- We use **greedy** method



Classification and Regression Tree (CART)

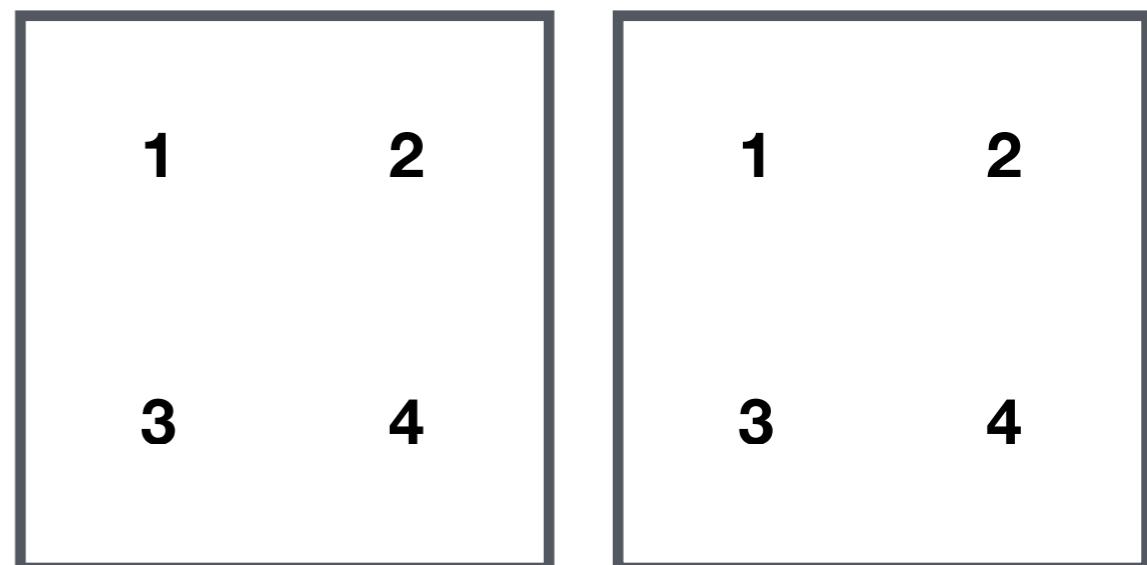
Greedy Algorithm: Look at one variable every time

Grow a tree by recursively repeating:

1. Pick a variable and the split-point which decreases $\hat{R}(f)$ the most
2. Split the node into two child nodes

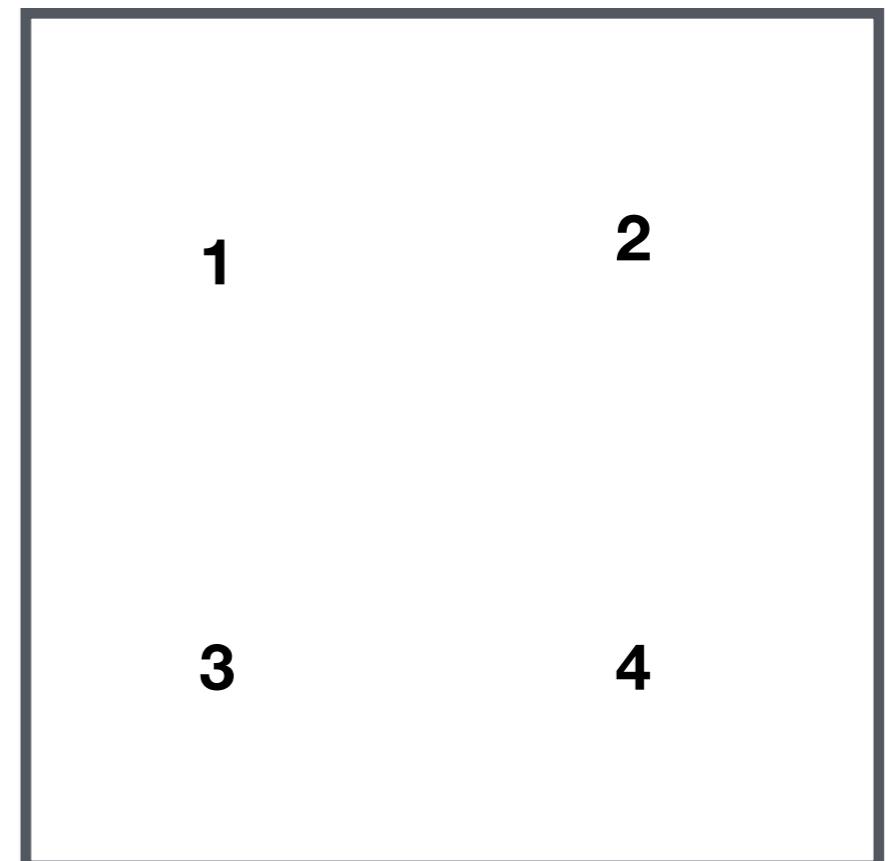
Until: each node has no more than K_{\min} samples

Example: Toy data



Classification and Regression Tree (CART)

Example: Toy data (continued)



CART: Tree Pruning

In practice, overfitting when K_{\min} is small

Idea: Regularize the tree size $|T| = \# \text{ of leaves}$

$$\min_T \hat{R}(\hat{f}_T) + \lambda|T|$$

Greedy Tree Pruning Method

1. Develop a fully grown tree T_0 with $K_{\min} = 1$
2. Collapse the tree T_0 by reversing the above process and get T_1, T_2, \dots
3. Pick one tree minimizing

$$\min_T \hat{R}(\hat{f}_T) + \lambda|T|$$

CART: Tree Pruning

Greedy Tree Pruning Method

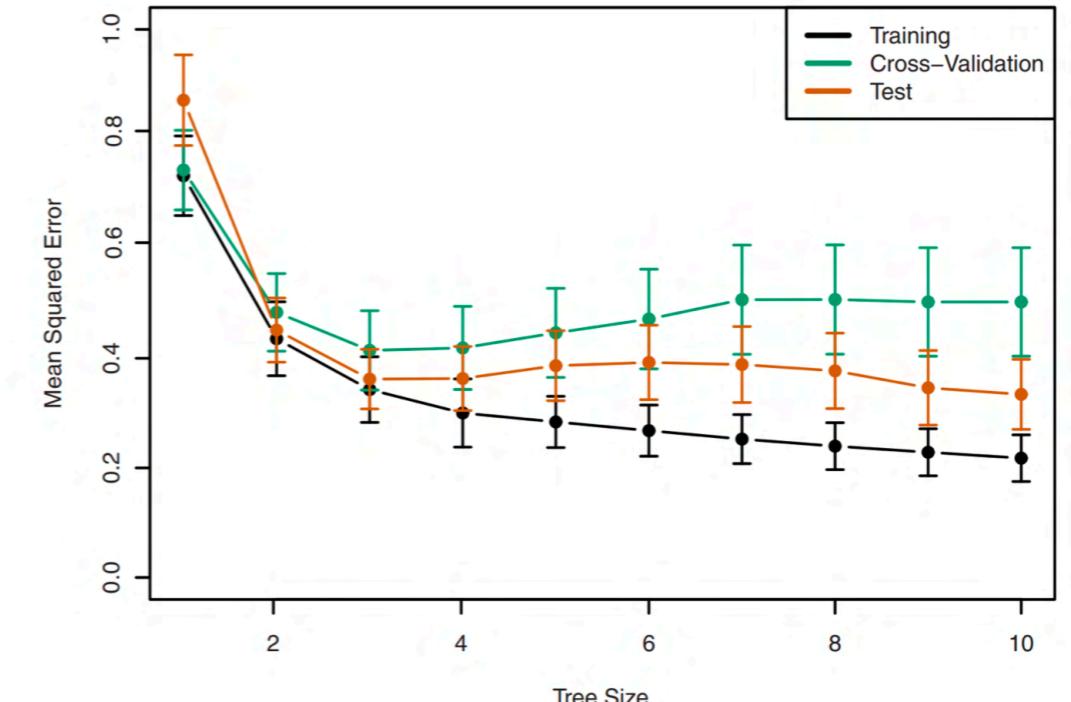
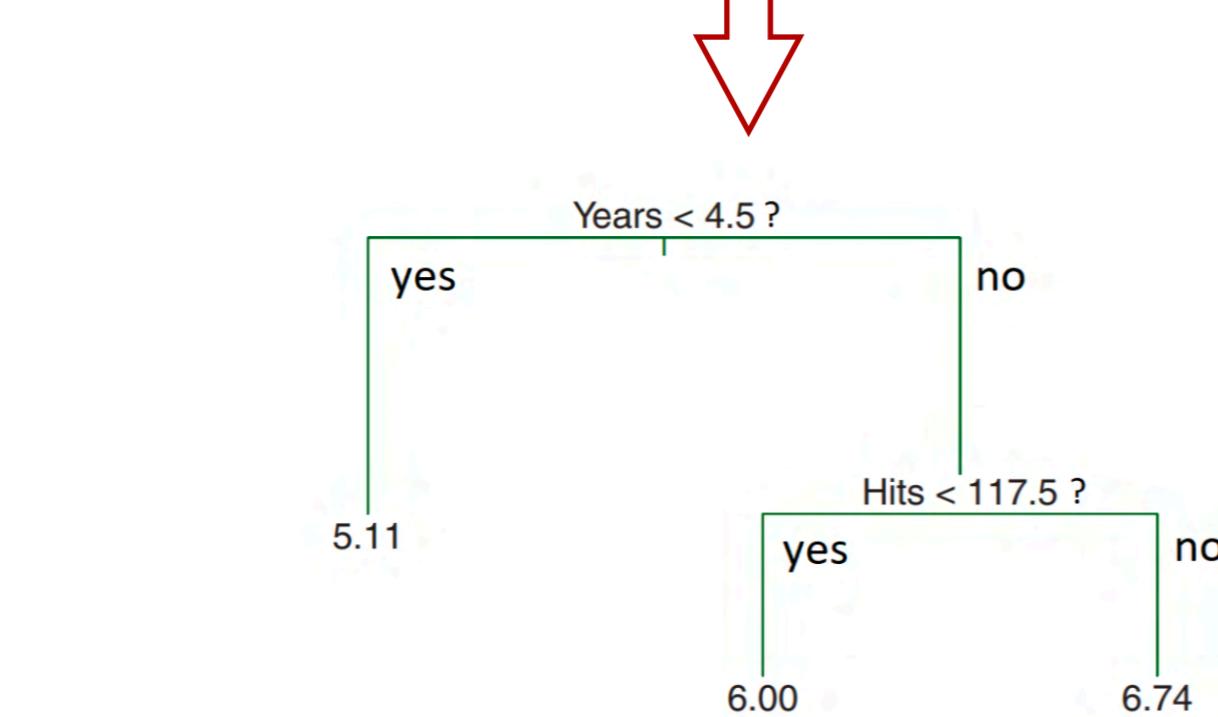
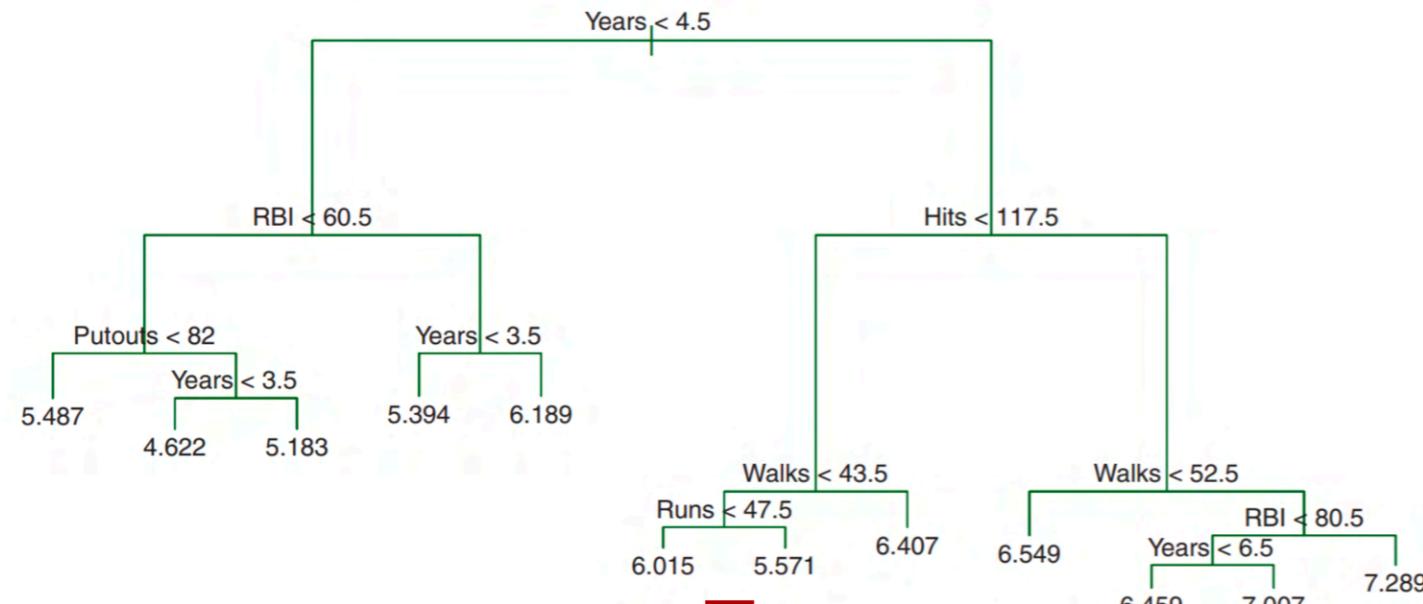
1. Develop a fully grown tree T_0 with $K_{\min} = 1$
2. Collapse the tree T_0 by reversing the above process and get T_1, T_2, \dots
3. Pick one tree minimizing

$$\min_T \hat{R}(\hat{f}_T) + \lambda|T|$$

1	2
3	4

CART: Tree Pruning

Example: Decide baseball play salaries



Classification and Regression Tree (CART)

CART can also be used for **classification**

Step 1. Determine a loss function

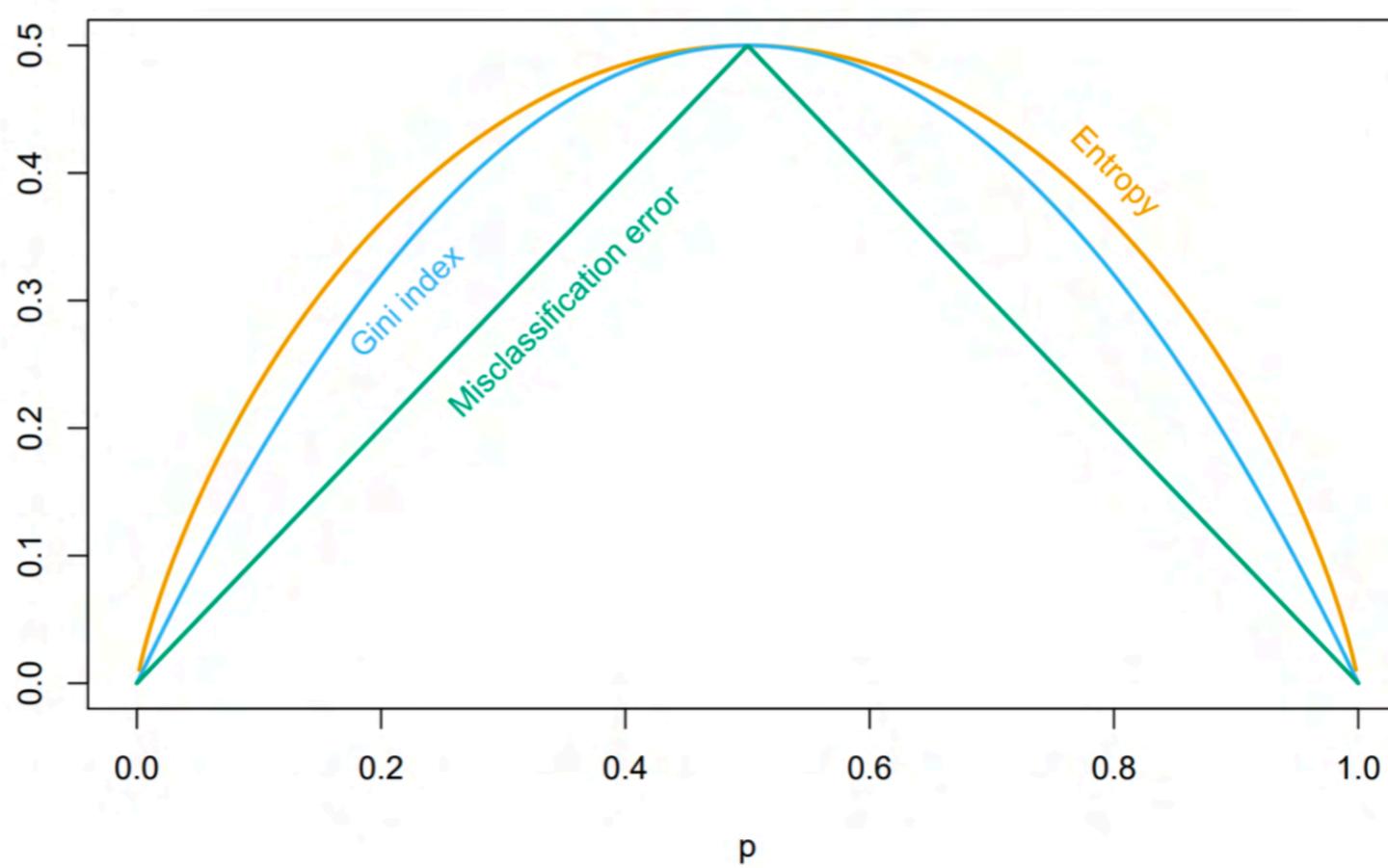
\hat{p}_{mk} = proportion of training points in region m that belong to class k :

$$\hat{p}_{mk} = \frac{1}{|R_m|} \sum_{i:x_i \in R_m} \mathbb{I}(y_i = k)$$

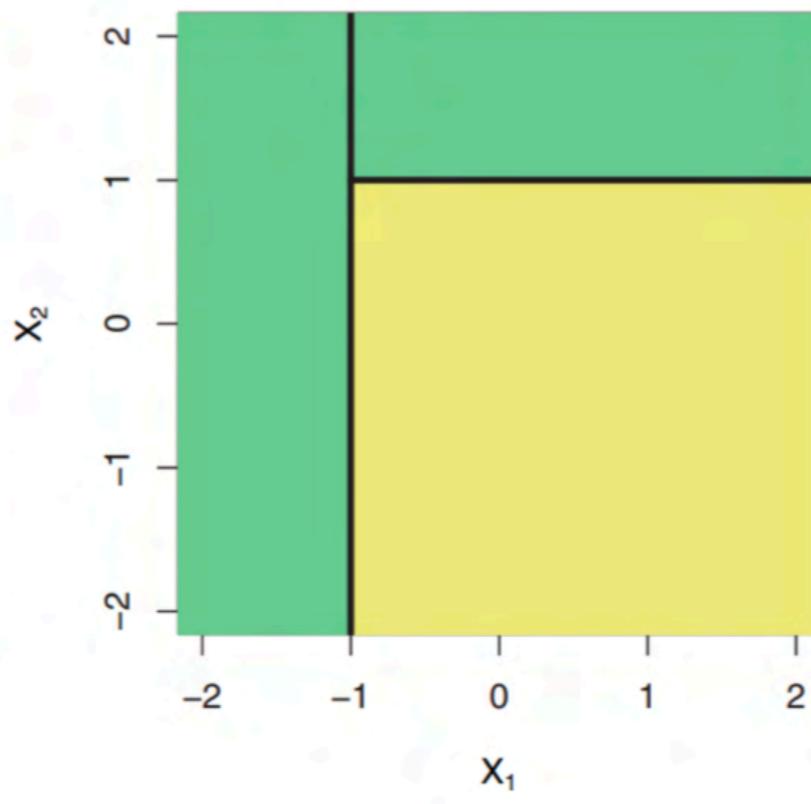
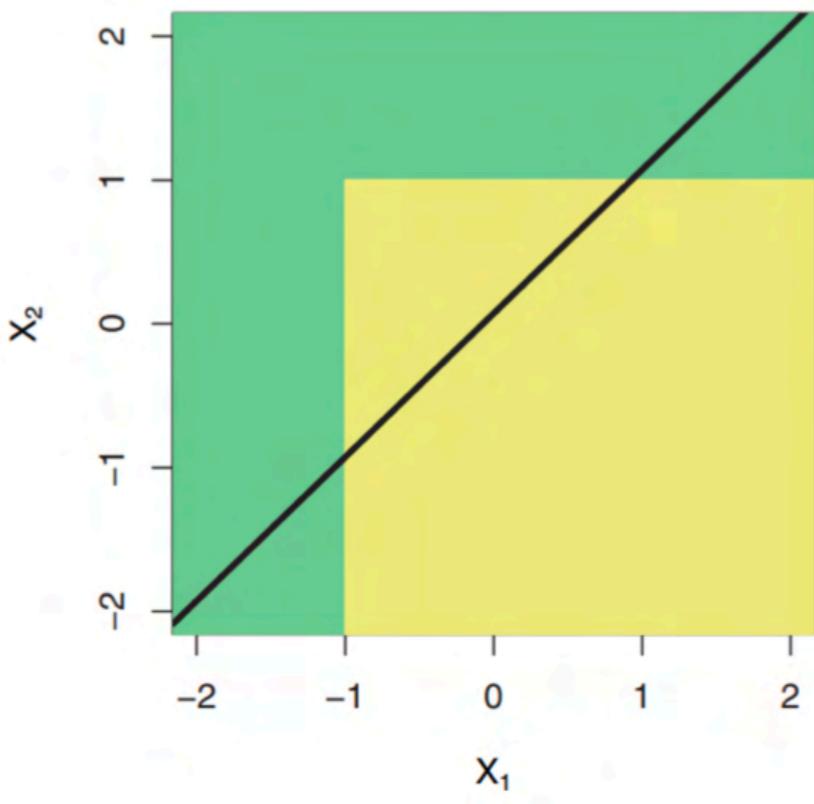
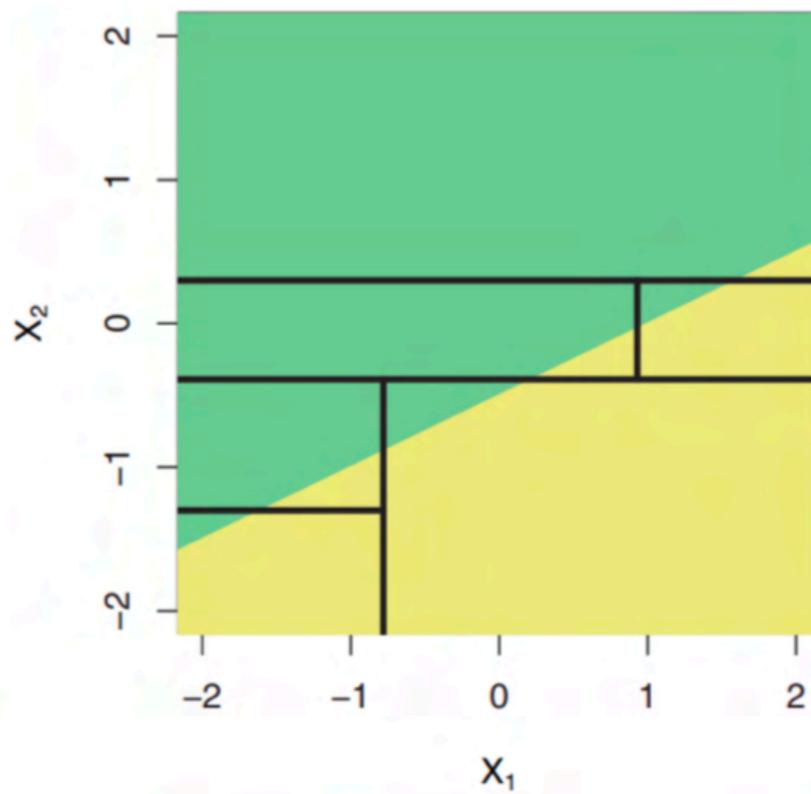
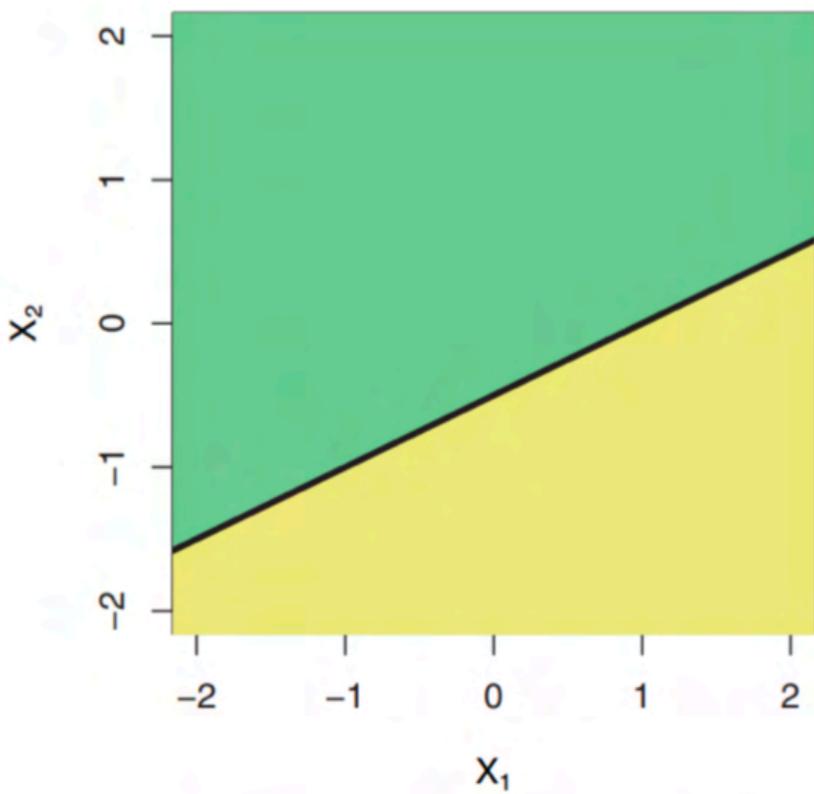
- **Missclassification error:** $E = \sum_{m=1}^M |R_m| \sum_k (1 - \max_k \hat{p}_{mk})$
 - **Gini error:** $G = \sum_{m=1}^M |R_m| \sum_k \hat{p}_{mk} (1 - \hat{p}_{mk})$
 - **Entropy:** $H = - \sum_{m=1}^M |R_m| \sum_k \hat{p}_{mk} \log \hat{p}_{mk}$
- Work better in practice**

Loss Functions for Binary Case

- **Missclassification error:** $E = \sum_{m=1}^M |R_m| \sum_k (1 - \max_k \hat{p}_{mk})$
 - **Gini error:** $G = \sum_{m=1}^M |R_m| \sum_k \hat{p}_{mk}(1 - \hat{p}_{mk})$
 - **Entropy:** $H = - \sum_{m=1}^M |R_m| \sum_k \hat{p}_{mk} \log \hat{p}_{mk}$
- Work better in practice**



Tree versus Linear Models



Pros/Cons of CART

Pros

- Interpretable, and easy to explain to non-experts
- Fast to train and fast for prediction
- Handle high dimensional data gracefully
- Handles collinear/dependent predictors with no problems

Cons

- Poor bias-variance trade-off
- Lack of smoothness

Tree Method

Junwei Lu



HARVARD
T.H. CHAN

SCHOOL OF PUBLIC HEALTH
Department of Biostatistics



Classification Review

$$P(Y, X) = P(Y|X) P(X)$$

↓
Model
↓
Ignore

● Discriminative classifier: Model $P(Y|X)$

Logistic regression $P(Y|X) = \frac{1}{1 + \exp(-yf(x))}$

● Generative classifier: Model $P(X|Y)$

$$P(Y, X) = \underbrace{P(X|Y)}_{\text{model both}} P(Y) \xrightarrow{\text{Ber}(p)}$$

LDA, QDA, Naive Bayes $\rightarrow P(X|Y)$ independent

$$P(X|Y) \sim \text{Gaussian}$$

● Model-free classifier

SVM, KNN \rightarrow We do not impose any distribution



Question: How to choose the method in practice?

Comparison of Classification Methods

n = number of samples, d = number of features

If $d \gg n$ High dimensional. e.g. $d = 1,000 \sim 100,000$ $n = 100 \sim 900$

Use logistic, linear SVM, LDA
regularization needed

Logistic depends on all training data, but SVM only
depends on support vectors (not all data)

} Measure performance
on test data.

If d is small, but n is intermediate e.g. $d = 10 \sim 1,000$ $n = 10 \sim 1,000$
Nonlinear model will be OK

Use SVM with Gaussian kernels, QDA

But they may overfit for HD data

If d is small, but n is large \rightarrow Massive: $d = 10 \sim 10,000$ $n \geq 100,000$
No worry for overfitting

Create more features, use logistic, linear SVM, LDA

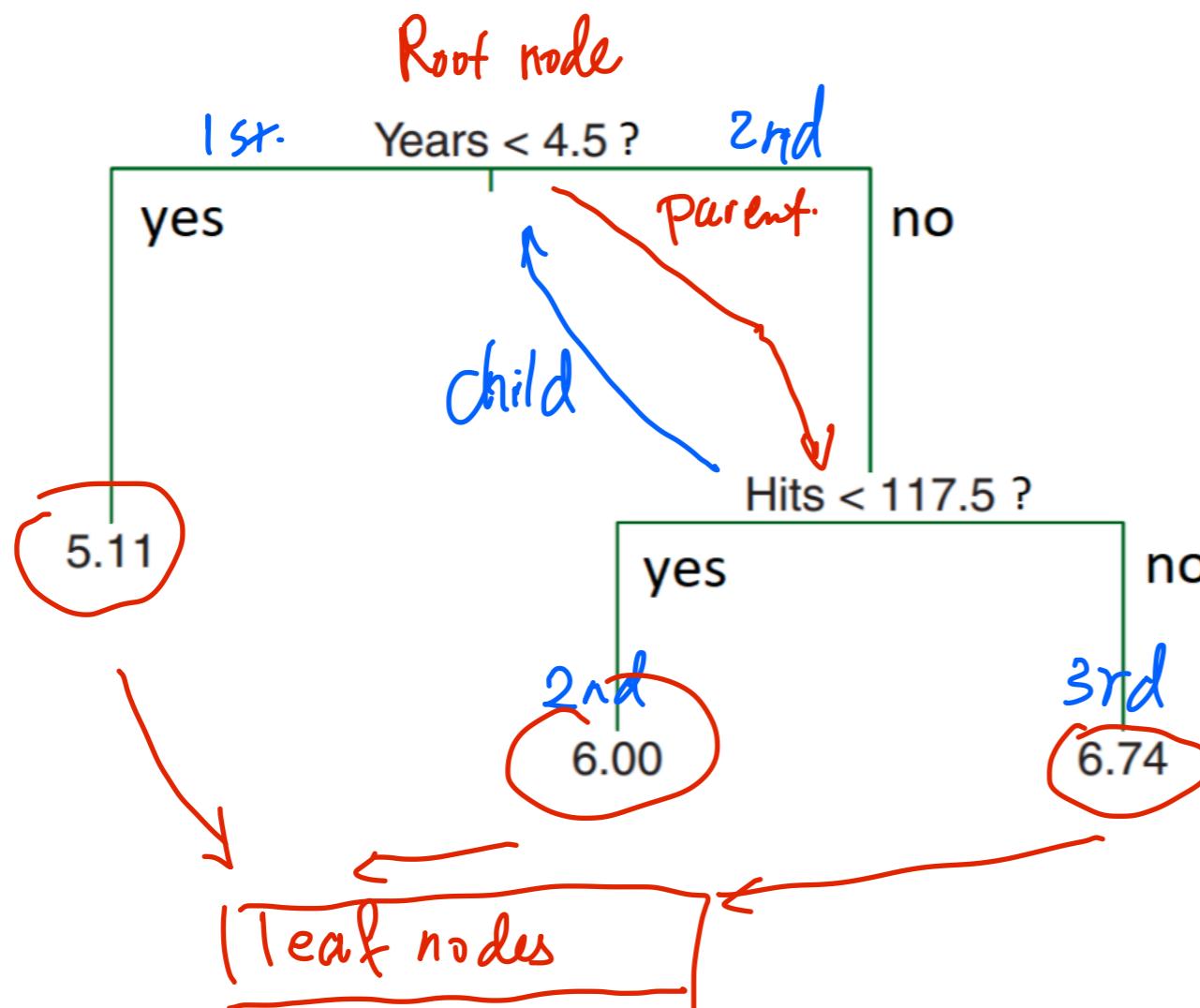
k-NN with no interoperability

In practice: Evaluate the performance on test data

Decision Tree

Decision tree is an **interoperable** method

Example: Decide baseball play ^{or} salaries



Data: Y_i : player's salary

X_i : Years , Hits

X_{i1} X_{i2} .

Input: player information.

Years & Hits

1st. Year < 4.5 \Rightarrow Salary = 5.11.

2nd. Year > 4.5 .

Hits < 117.5 \Rightarrow Salary = 6.00

3rd

Year > 4.5

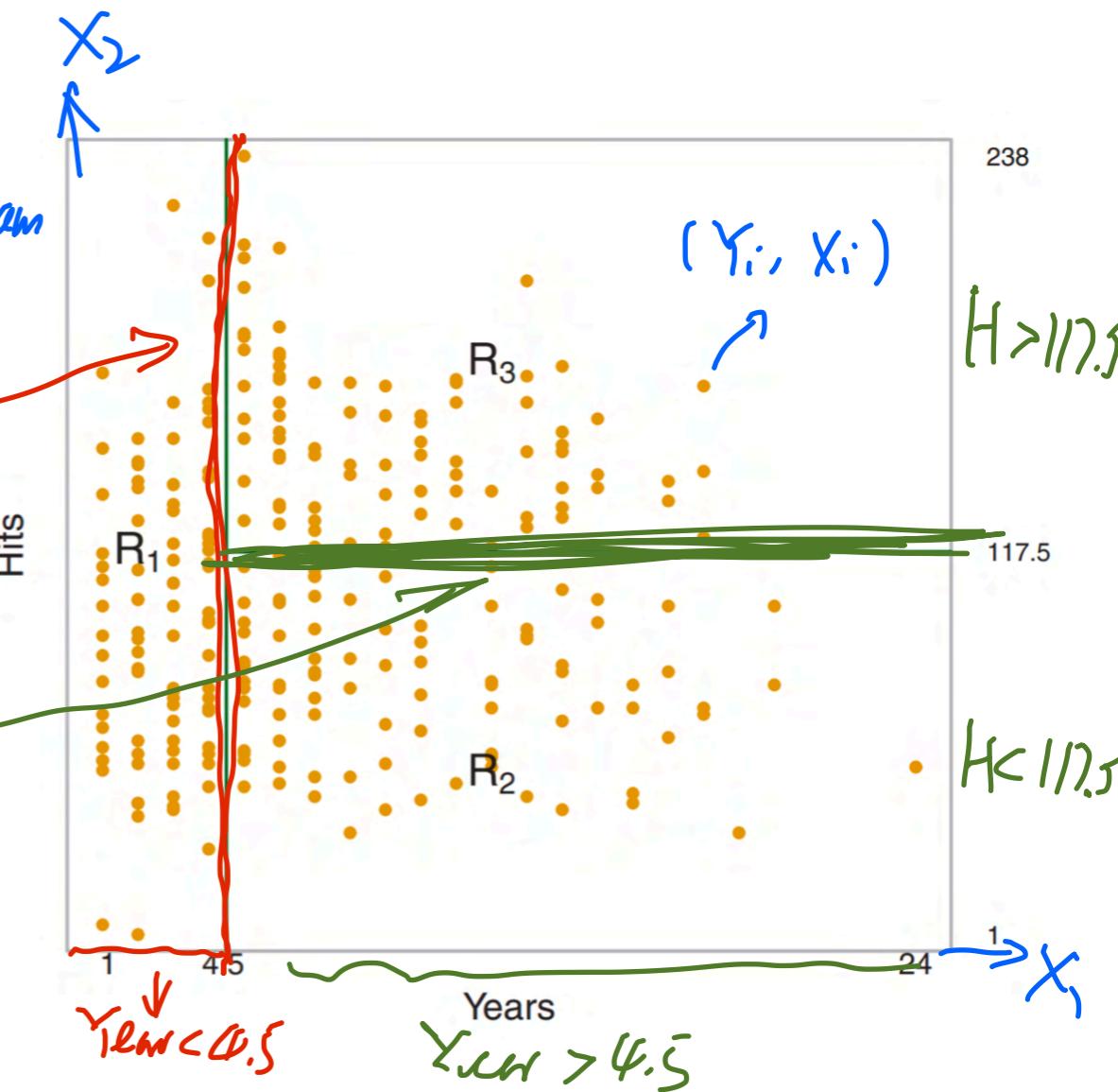
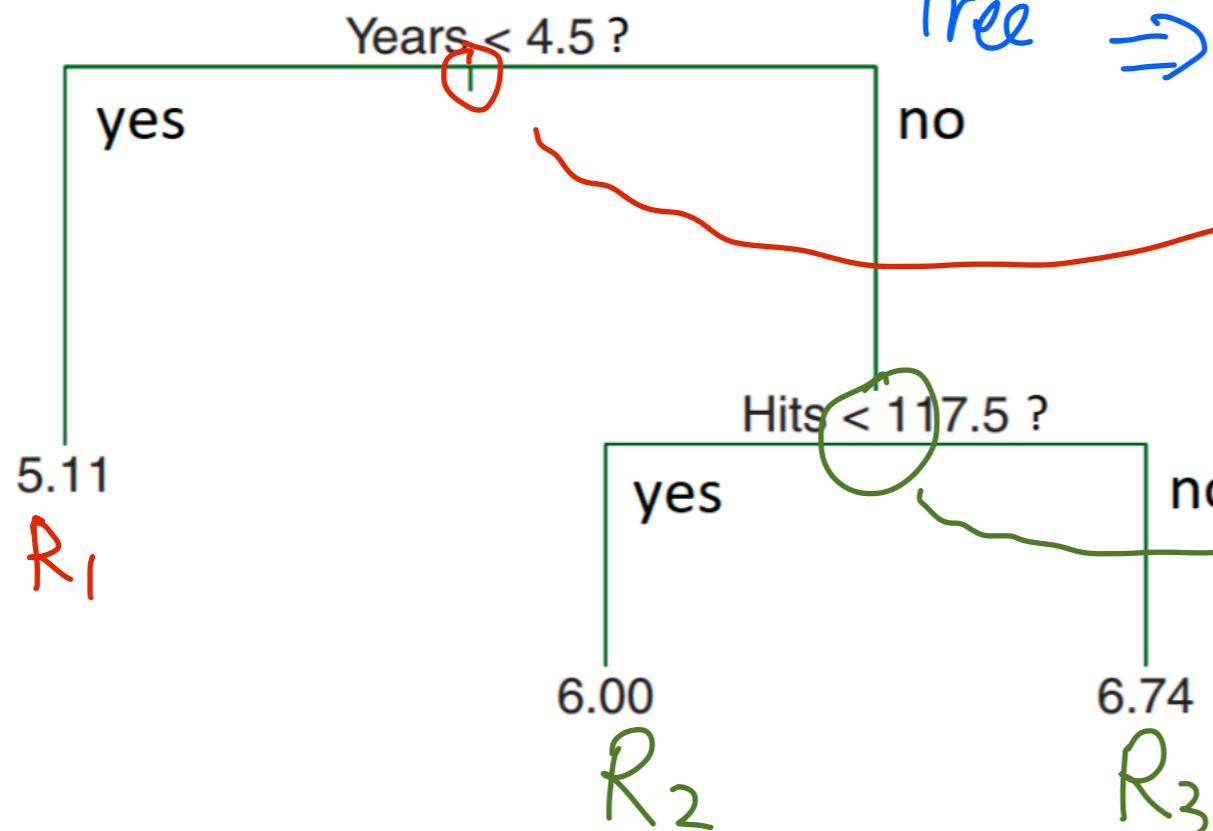
Hits > 117.5 \Rightarrow Salary = 6.74

Decision Tree

One-to-one map: Node \leftrightarrow Region

Decision tree is an ~~interoperable~~ method
interpretable

Example: Decide baseball play salaries



Concepts: Tree, leaf, node, child

Decision Tree

Train data $(Y_1, X_1), \dots, (Y_n, X_n)$

Tree leaf nodes \longleftrightarrow Regions

Q: How to fit the model using the tree?

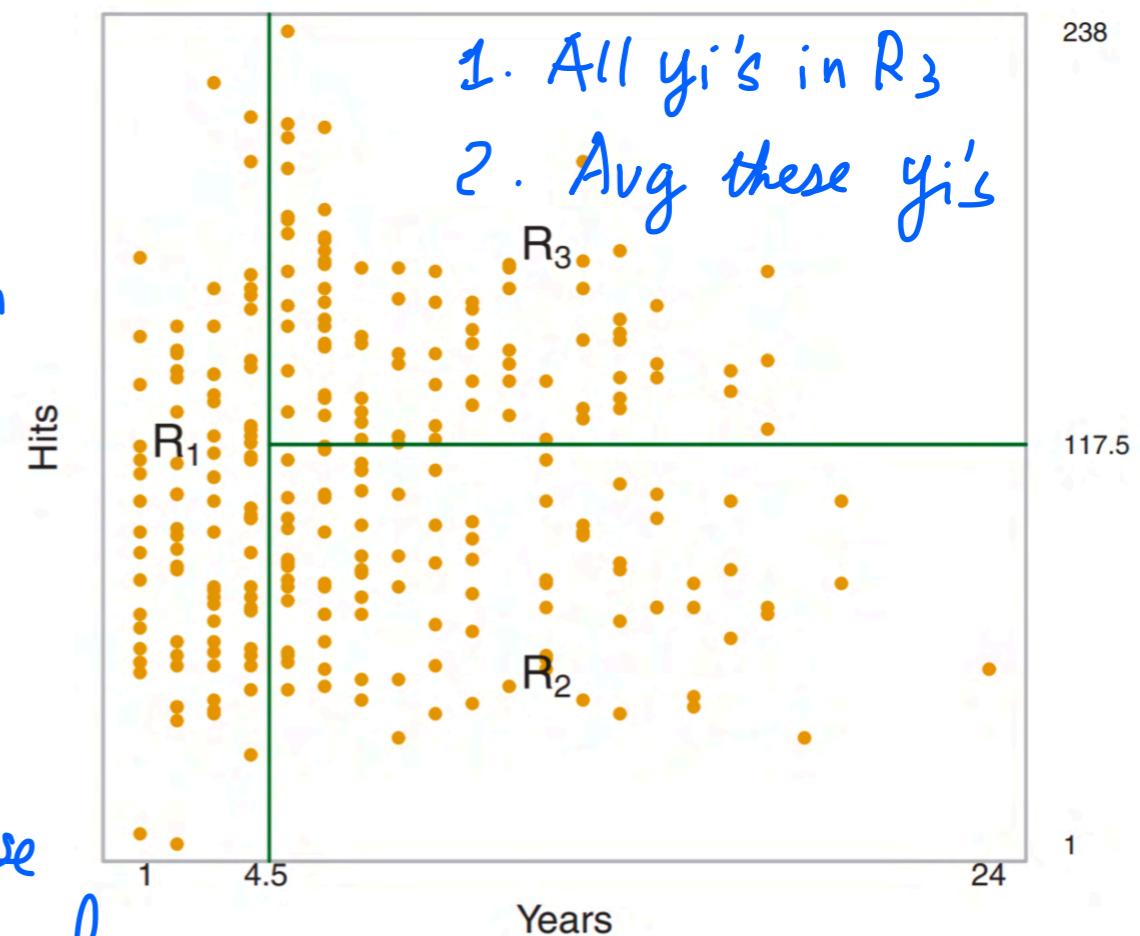
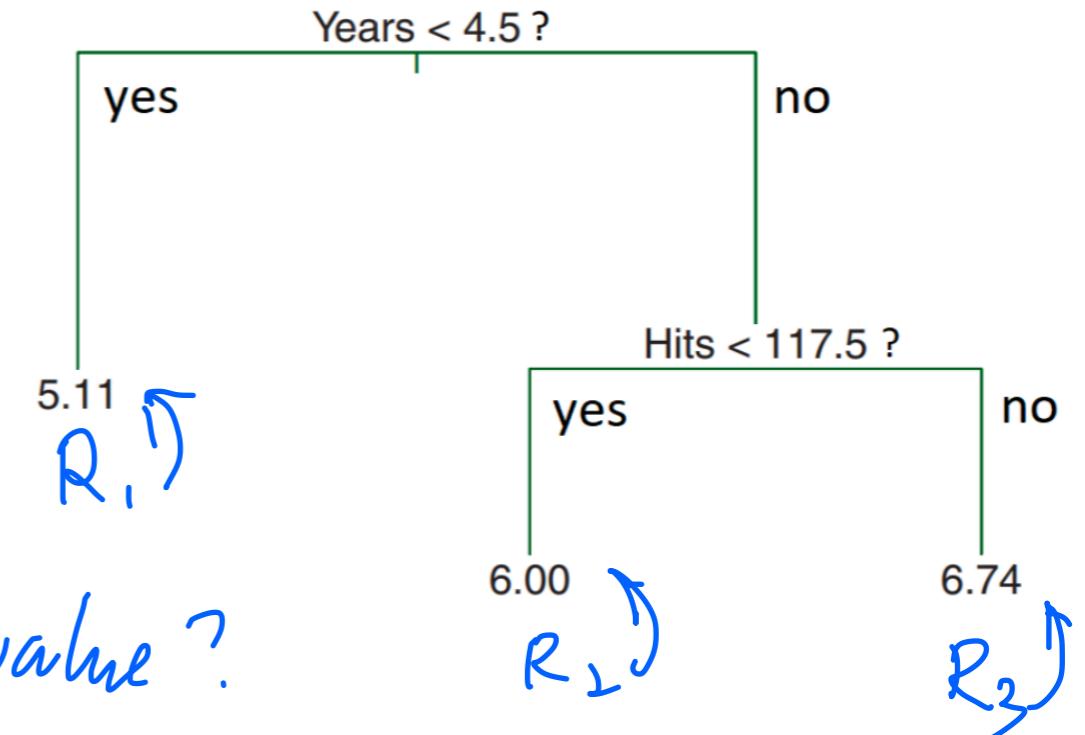
Given regions R_1, \dots, R_n . How to specify value?

For test point x , if $\underline{x} \in R_m$, predict $\hat{y} = \bar{y}_{R_m}$

$$\bar{y}_{R_m} = \frac{1}{|R_m|} \sum_{i: x_i \in R_m} y_i \Rightarrow \text{Avg in } R_m$$

Predict the average of the outcome in the region

$$\hat{f}(x) = \sum_{m=1}^M \bar{y}_{R_m} \mathbb{I}(x \in R_m) \Rightarrow \text{Piecewise constant function.}$$

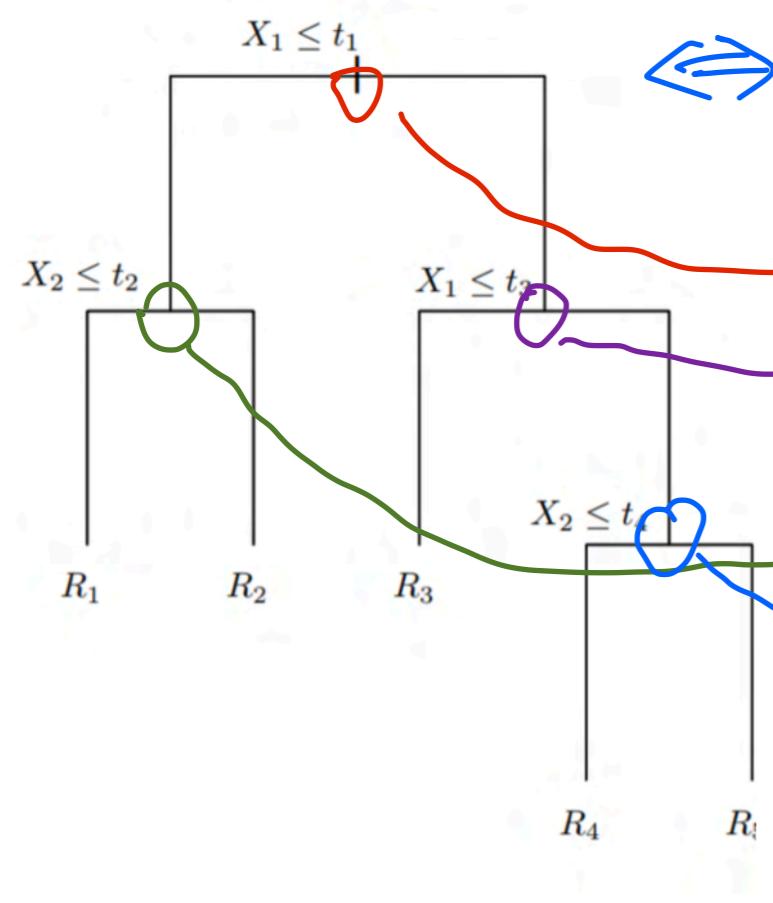


Decision Tree

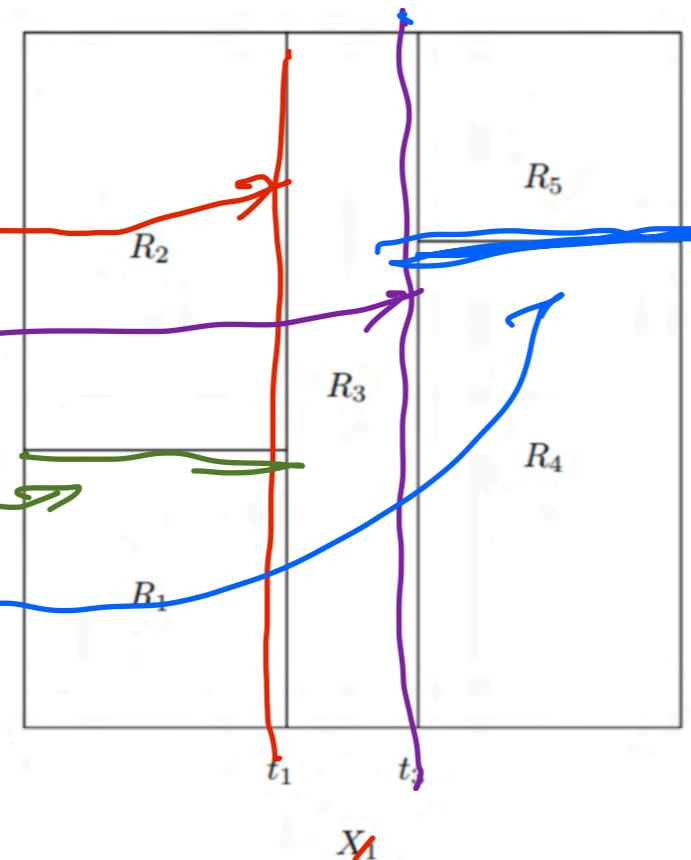
The fitted function is piecewise constant

$$\mathcal{F} = \left\{ f(x) = \sum_{m=1}^M \bar{y}_{R_m} \mathbb{I}(x \in R_m) \right\}$$

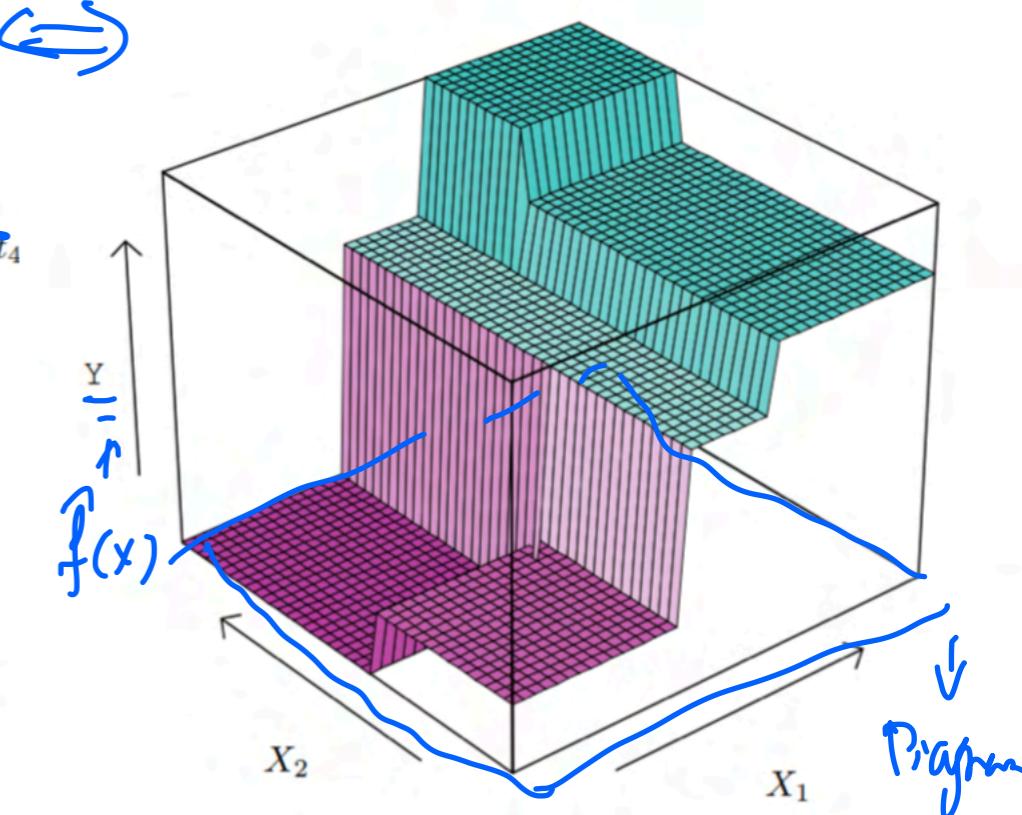
Tree



Diagram

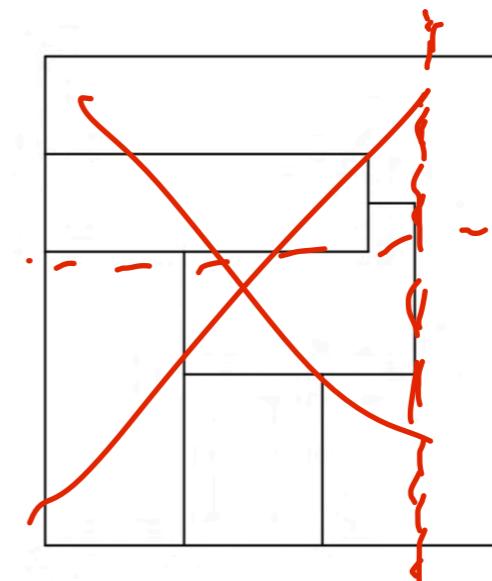
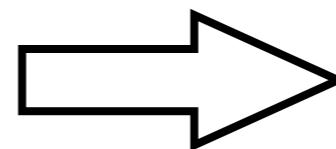


Fitted function



Diagram

This image is impossible



Tree-based Regression Method



Question: How do we learn the tree from training data?
Regions & Y's

Question: How to split the diagram into regions?

Step 1. Determine a loss function, e.g., ℓ_2 -loss: $R(f) = \mathbb{E}[Y - f(X)]^2$

Empirical ℓ_2 -loss $\Rightarrow \hat{R}(f) = \frac{1}{n} \sum_{i=1}^n (Y_i - f(X_i))^2$

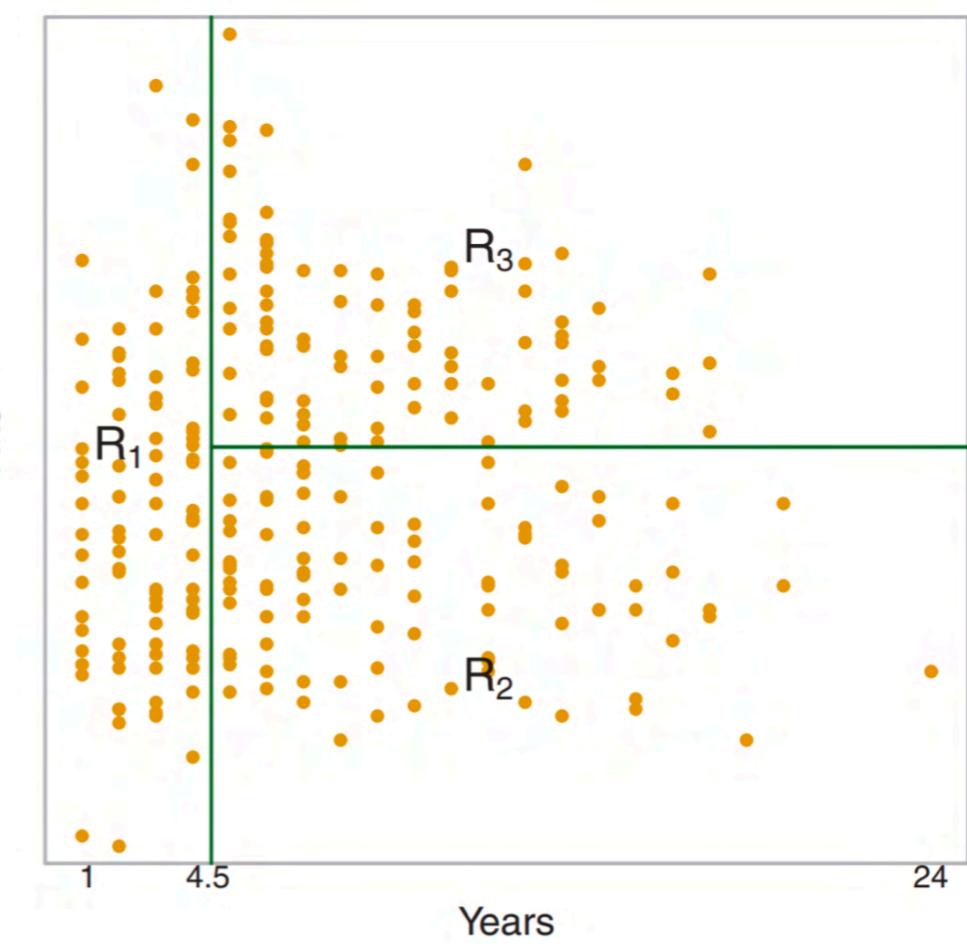
Since $f(x) = \sum_{m=1}^M \bar{y}_{R_m} \mathbb{I}(x \in R_m)$

Plug f above
into $\hat{R}(f) \Rightarrow \hat{R}(f) = \frac{1}{n} \sum_{m=1}^M \sum_{i:x_i \in R_m} (Y_i - \bar{y}_{R_m})^2$

Sum over regions in each region

$= \sum_{m=1}^M \ell_2\text{-loss in region } R_m$

→ Piecewise constant



Classification and Regression Tree (CART)



Question: How do we learn the tree from training data?

Step 1. Determine a loss function, e.g., ℓ_2 -loss: $R(f) = \mathbb{E}[Y - f(X)]^2$

$$\hat{R}(f) = \frac{1}{n} \sum_{m=1}^M \sum_{i:x_i \in R_m} (Y_i - \bar{y}_{R_m})^2$$

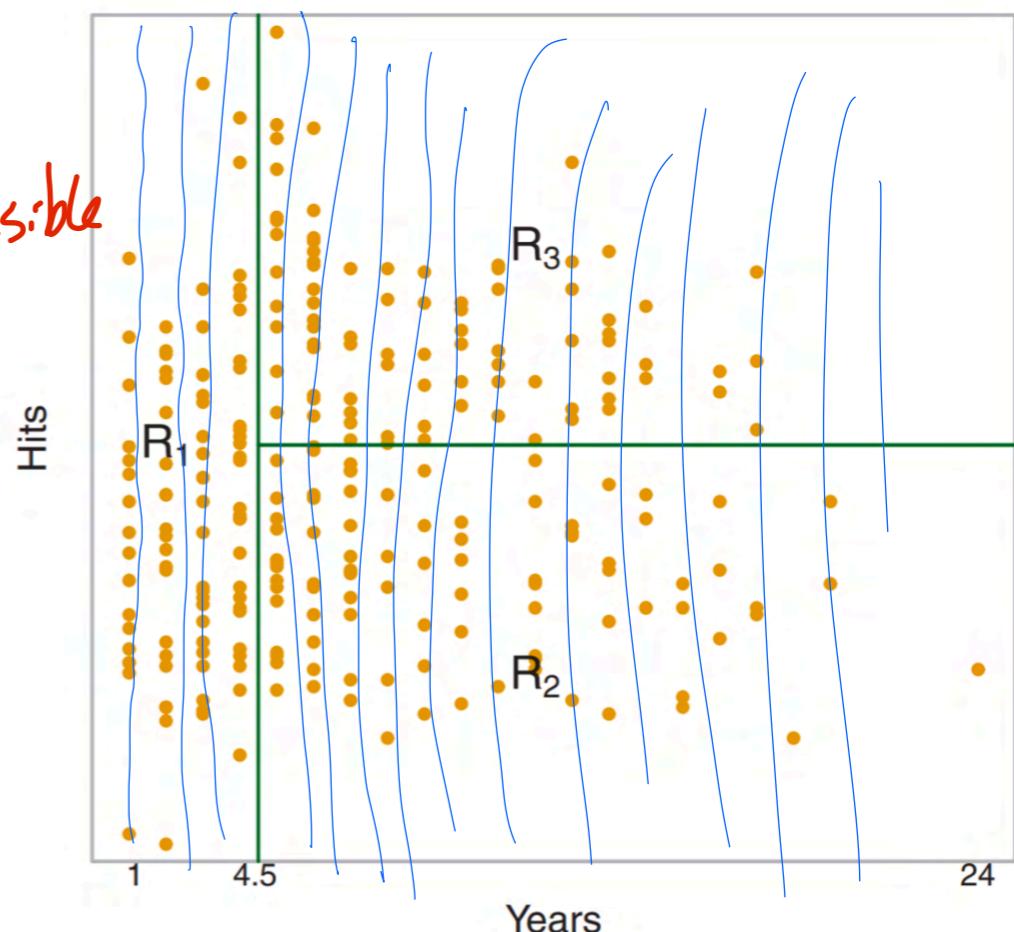
fast in practice
Optim 1 : Var X_i , Possible splits : n split

Step 2. Find the good “tree partition”

- Intractable to find the best tree
- We use **greedy** method

to find a good cut.

Many possible cuts



Classification and Regression Tree (CART)

Greedy Algorithm: Look at one variable every time

Grow a tree by recursively repeating:

1. Pick a variable and the split-point which decreases $\hat{R}(f)$ the most

One of X_1, X_2, \dots, X_d

$X_j < t$ or $X_j \geq t \Rightarrow Q: \text{How to Specify } t$

Since t continuous.

2. Split the node into two child nodes

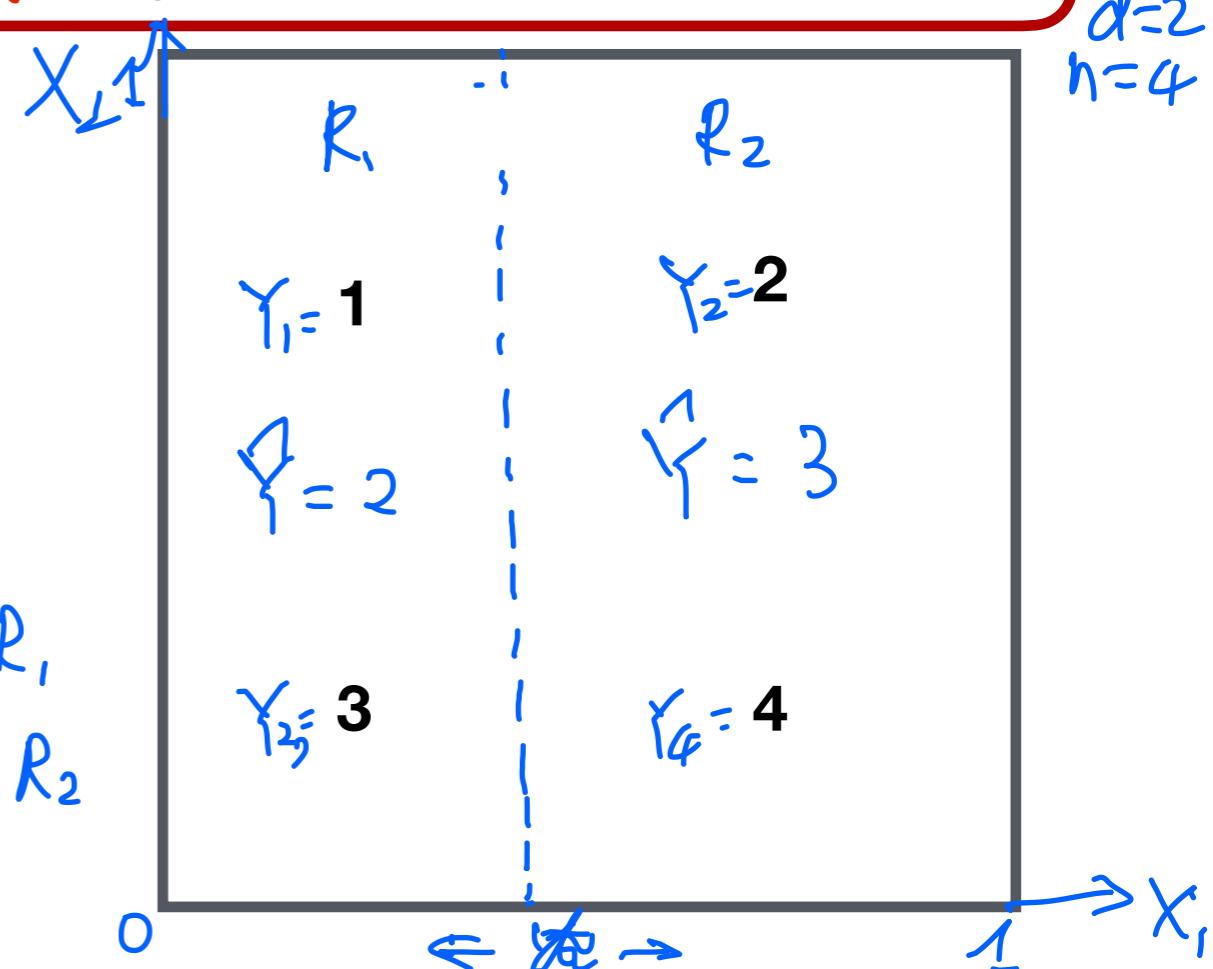
Until: each node has no more than K_{\min} samples

Example: Toy data ($K_{\min} = 1$)

Option 1: Variable X_1 , $X_1 < t = V_2$

Calculate $\hat{f} = \begin{cases} 2 & \text{if } X \text{ in } R_1 \\ 3 & \text{if } X \text{ in } R_2 \end{cases}$

$$\begin{aligned}\hat{R}(\hat{f}) &= (1-2)^2 + (3-2)^2 \quad \text{l}_2 \text{ loss in } R_1 \\ &\quad + (2-3)^2 + (4-3)^2 \quad \text{l}_2 \text{ loss in } R_2 \\ &= 4\end{aligned}$$



Classification and Regression Tree (CART)

Example: Toy data (continued)

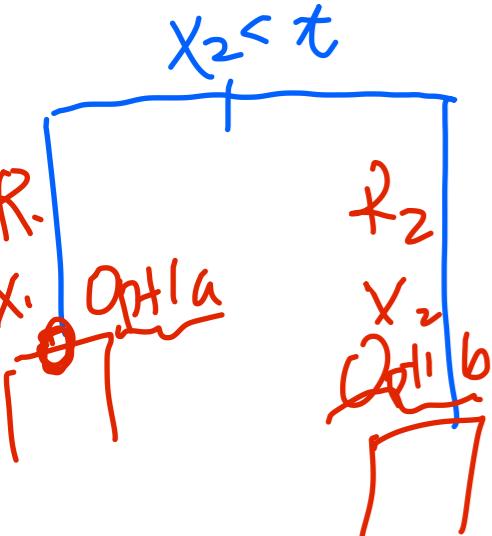
Option 2: Variable X_2 . Split $X_2 < t$,

$$\hat{f}(x) = \begin{cases} 1.5 & x \in R_1 \\ 3.5 & x \in R_2 \end{cases}$$

$$\begin{aligned}\hat{R}(\hat{f}) &= (1 - 1.5)^2 + (2 - 1.5)^2 \quad \text{l_2-loss in } R_1 \\ &\quad + (3 - 3.5)^2 + (4 - 3.5)^2 \quad \text{l_2-loss in } R_2 \\ &= \frac{1}{4} \times 4 = 1\end{aligned}$$

Cut using X_2 is better than X_1

Pick X_2



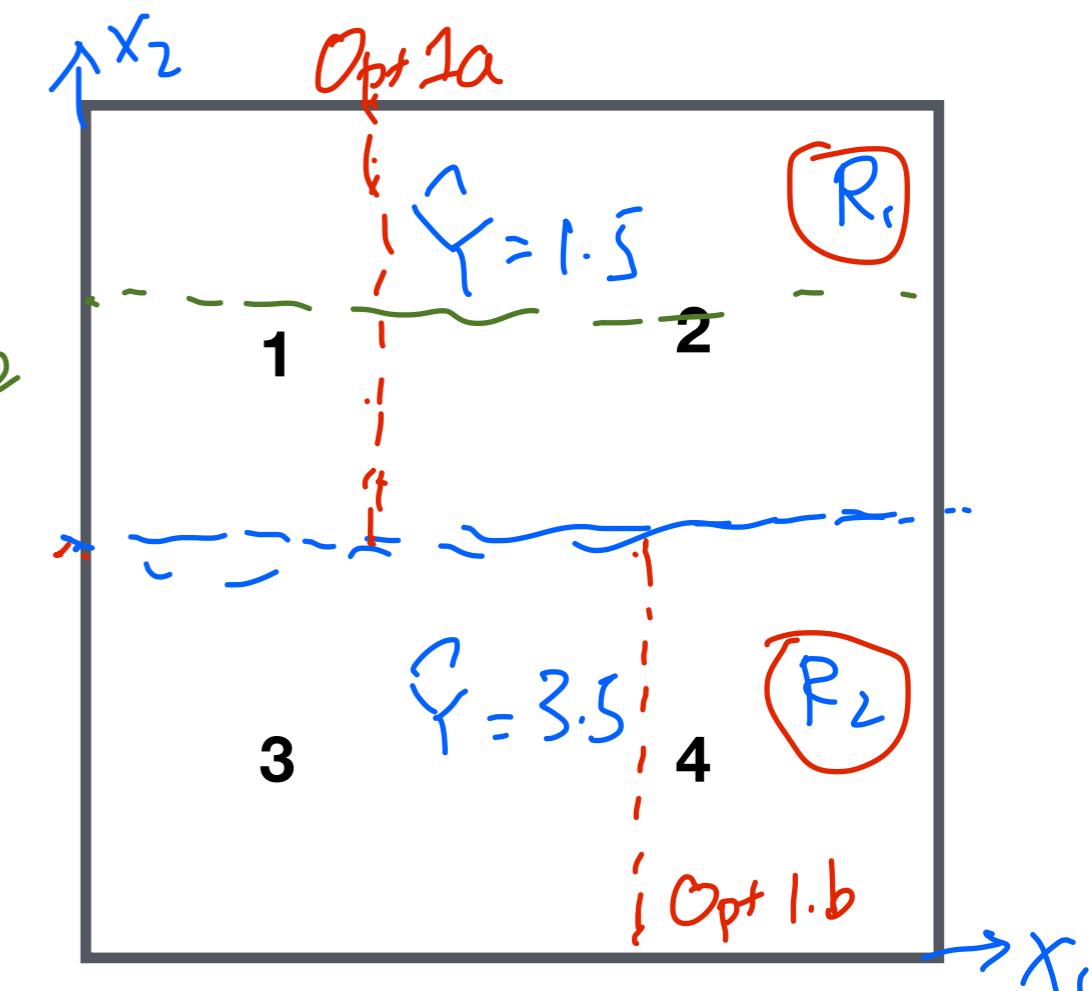
\Rightarrow Step 2: Split R_1 & R_2

Option 1. Var X_1 vs Var X_2



Cut X_1 for both

Opt 1a & 1b

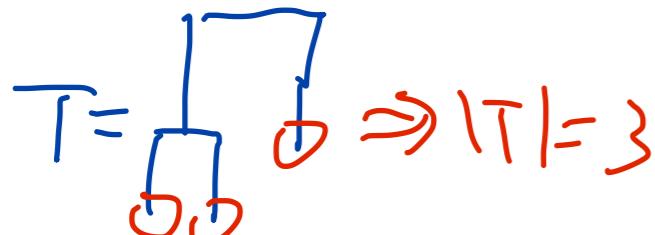


CART: Tree Pruning

In practice, overfitting when K_{\min} is small

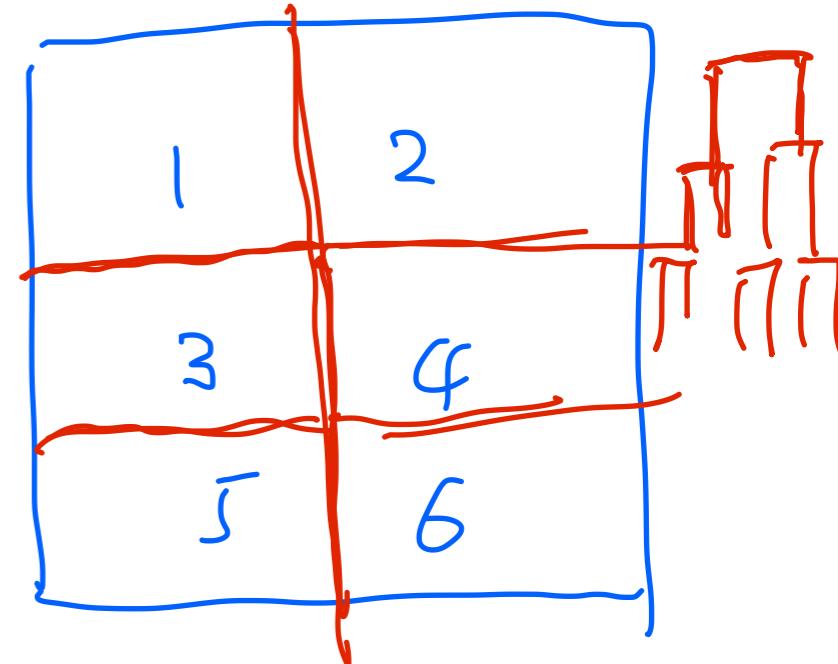
$K_{\min} = 1$
overfits!

Idea: Regularize the tree size $\underline{|T|} = \# \text{ of leaves}$



$$\min_T \hat{R}(\hat{f}_T) + \lambda |T|$$

loss penalty
small



Greedy Tree Pruning Method

1. Develop a fully grown tree T_0 with $K_{\min} = 1$

2. Collapse the tree T_0 by reversing the above process and get T_1, T_2, \dots

3. Pick one tree minimizing

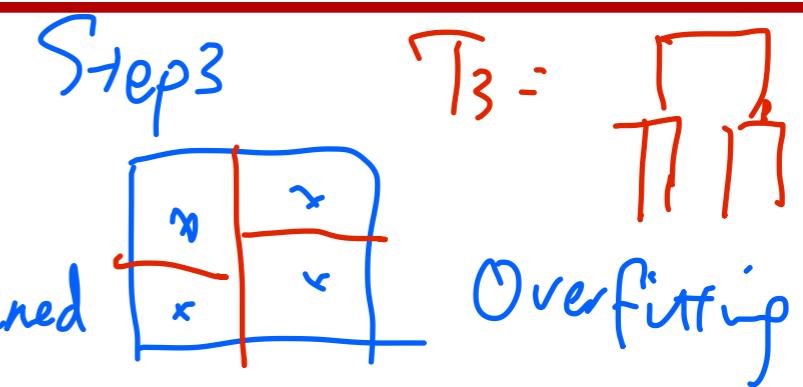
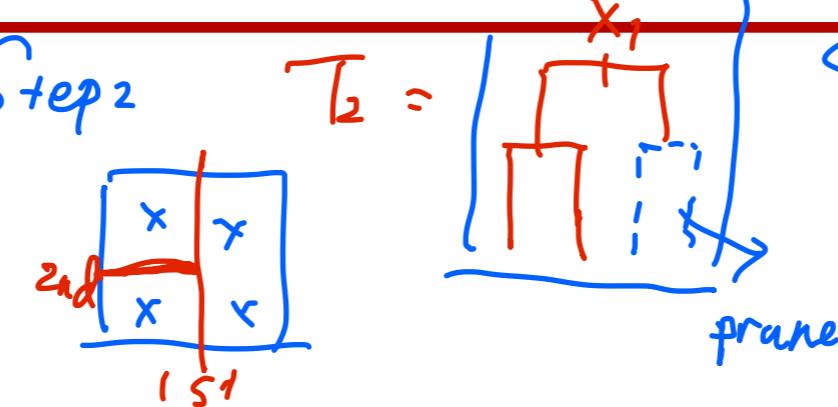
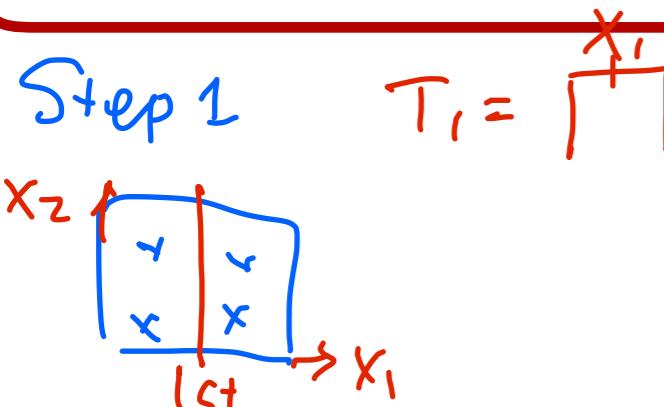
$$\min_T \hat{R}(\hat{f}_T) + \lambda |T|$$

$\ell(T)$

Calculate: $\ell(T_3), \ell(T_2), \ell(T_1)$

Find which minimize.

$$\hat{T} = \arg \min_T \ell(T)$$

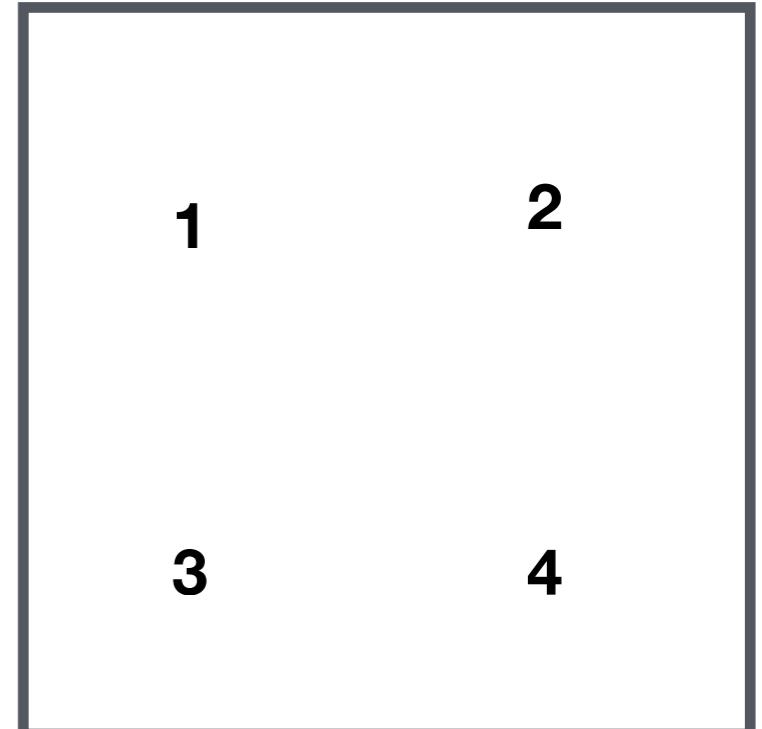
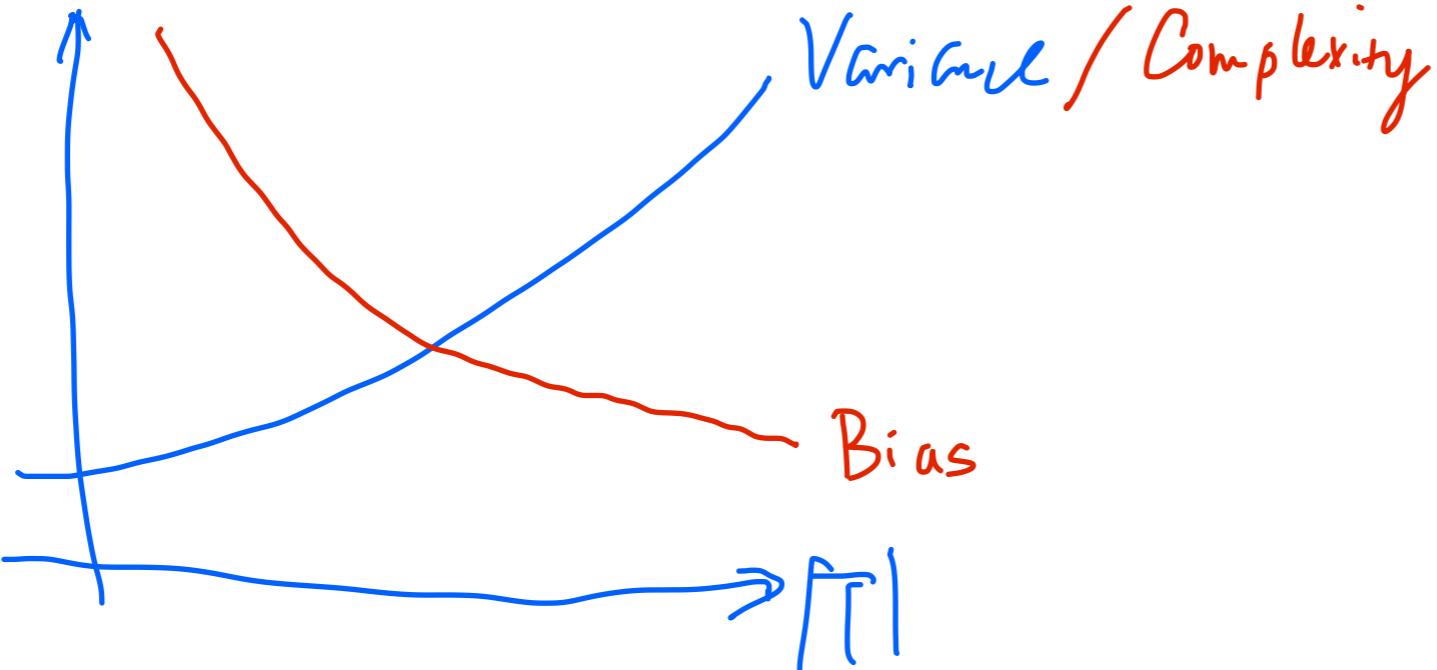


CART: Tree Pruning

Greedy Tree Pruning Method

1. Develop a fully grown tree T_0 with $K_{\min} = 1$
2. Collapse the tree T_0 by reversing the above process and get T_1, T_2, \dots
3. Pick one tree minimizing

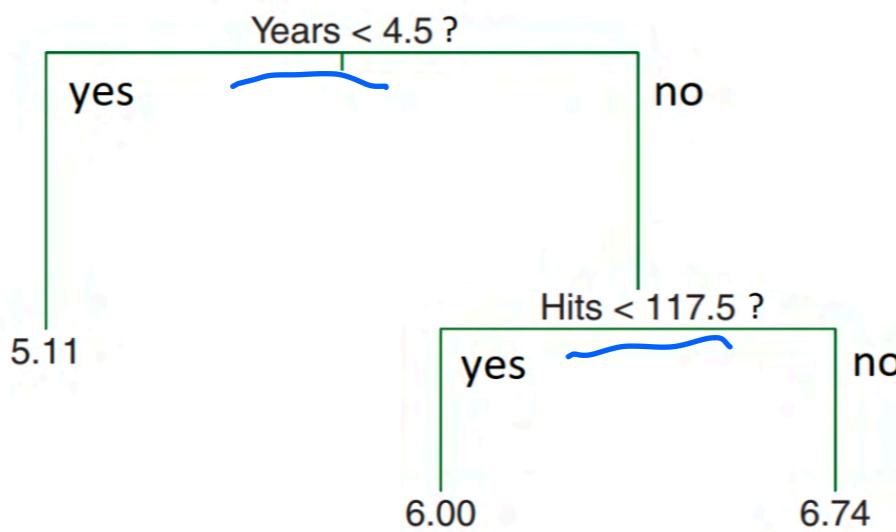
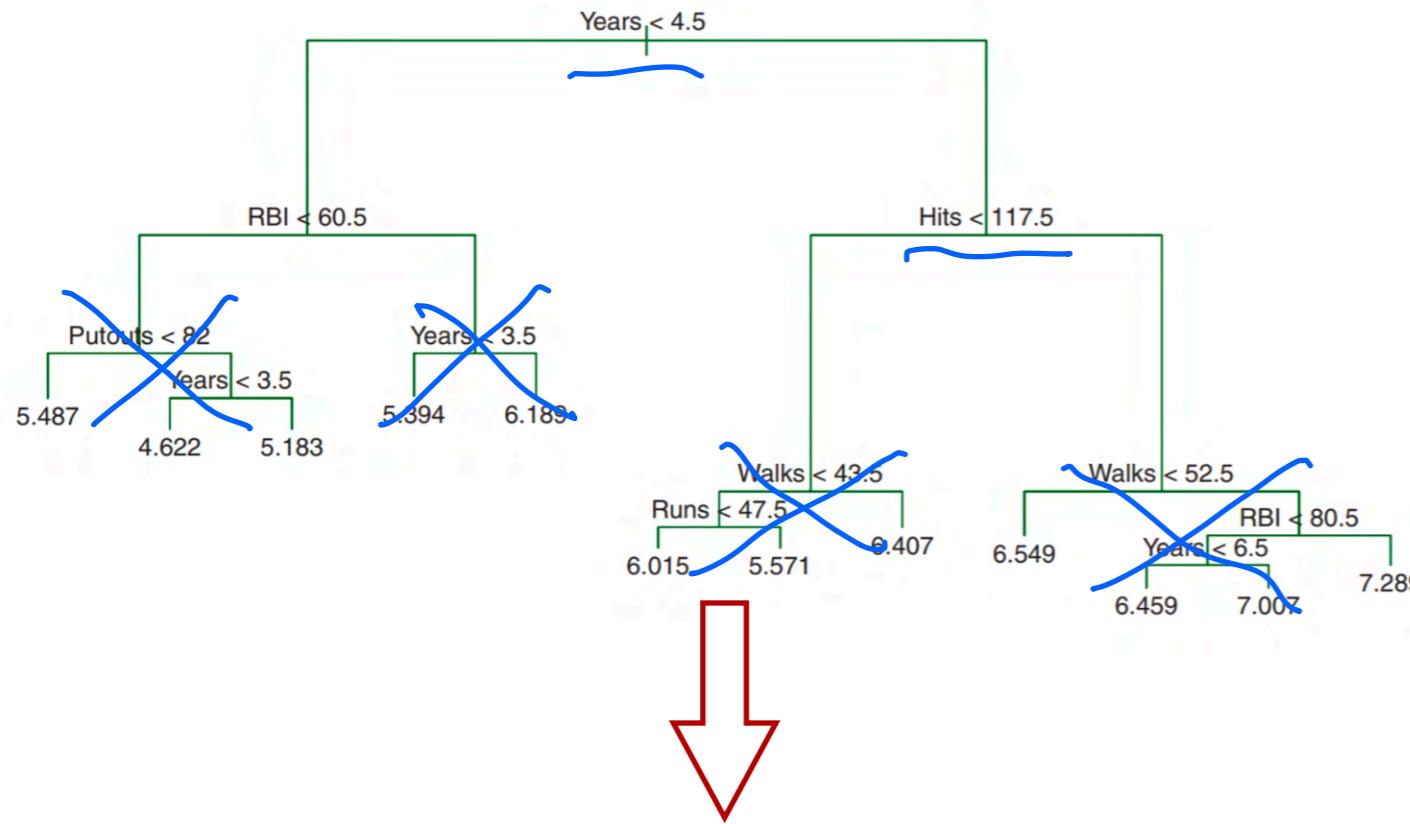
$$\min_T \hat{R}(\hat{f}_T) + \lambda|T|$$



CART: Tree Pruning

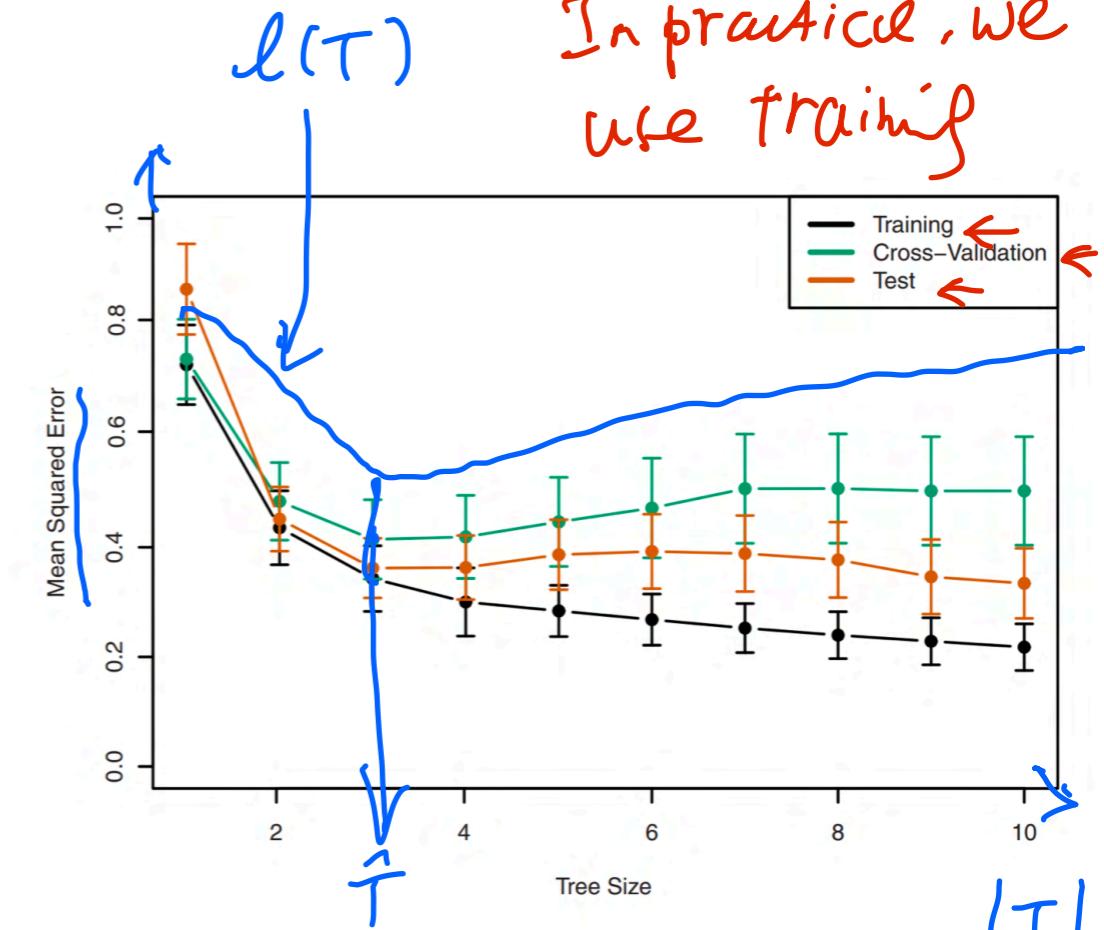
Example: Decide baseball play salaries

player example



$$l(T) = \underbrace{\text{MSE}}_{\text{Training}} + \lambda |T|$$

In practice, we use training



Simpler model to avoid overfitting!

Classification and Regression Tree (CART)

CART can also be used for classification

Step 1. Determine a loss function

\hat{p}_{mk} = proportion of training points in region m that belong to class k :

$$\hat{p}_{mk} = \frac{1}{|R_m|} \sum_{i:x_i \in R_m} \mathbb{I}(y_i = k) \quad \leftarrow \text{proportion of correctness}$$

losses
for
classification

- Missclassification error: $E = \sum_{m=1}^M |R_m| \sum_k (1 - \max_k \hat{p}_{mk})$ *not good in practice*
 - Gini error: $G = \sum_{m=1}^M |R_m| \sum_k \hat{p}_{mk} (1 - \hat{p}_{mk})$
 - Entropy: $H = - \sum_{m=1}^M |R_m| \sum_k \hat{p}_{mk} \log \hat{p}_{mk}$
- Work better in practice**

Loss Functions for Binary Case

Not smooth \Rightarrow Bad

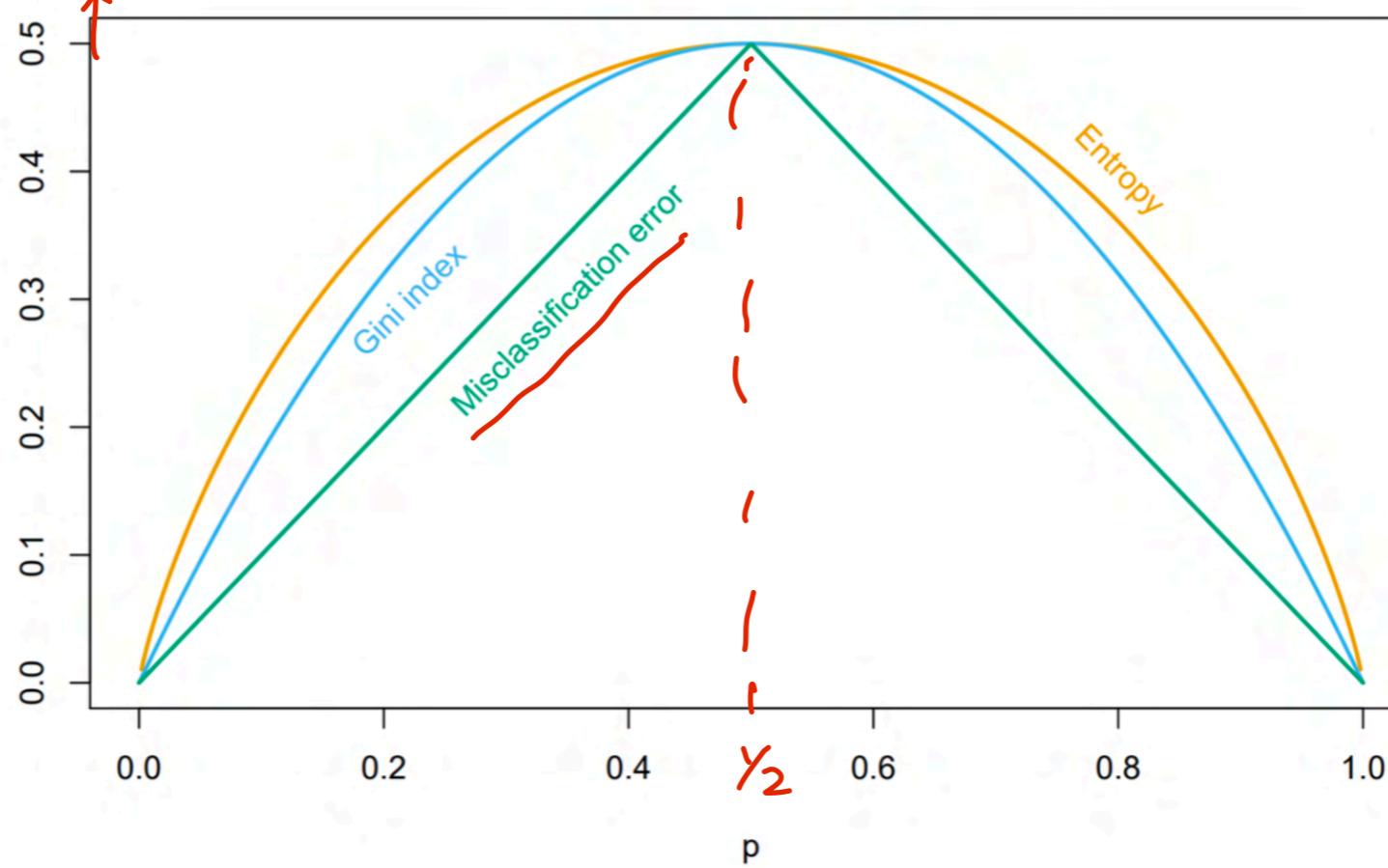
- **Missclassification error:** $E = \sum_{m=1}^M |R_m| \sum_k (1 - \max_k \hat{p}_{mk})$

- **Gini error:** $G = \sum_{m=1}^M |R_m| \sum_k \hat{p}_{mk} (1 - \hat{p}_{mk})$

- **Entropy:** $H = - \sum_{m=1}^M |R_m| \sum_k \hat{p}_{mk} \log \hat{p}_{mk}$

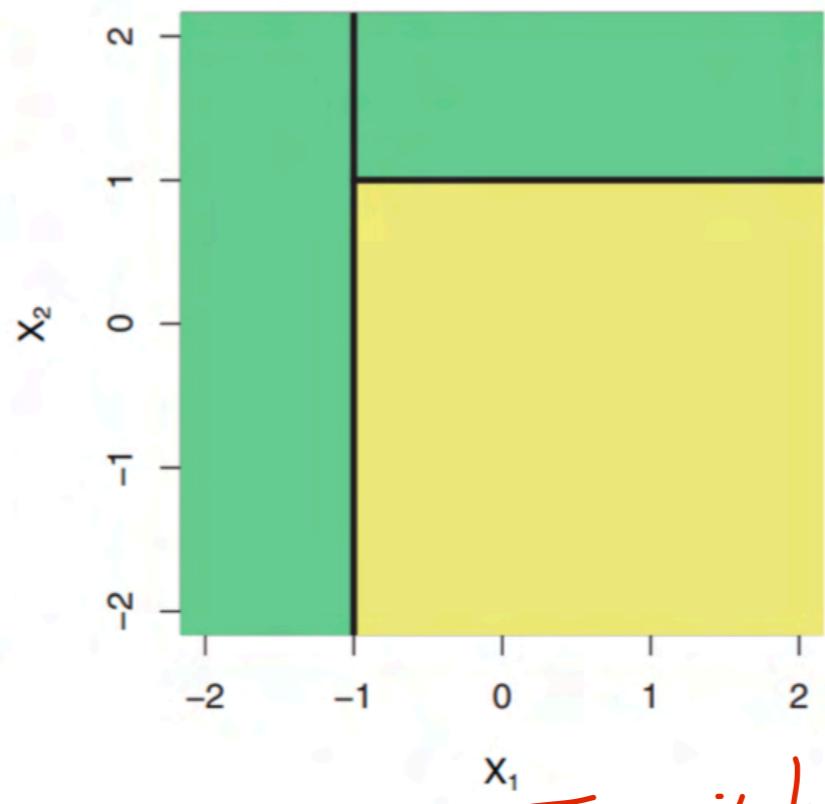
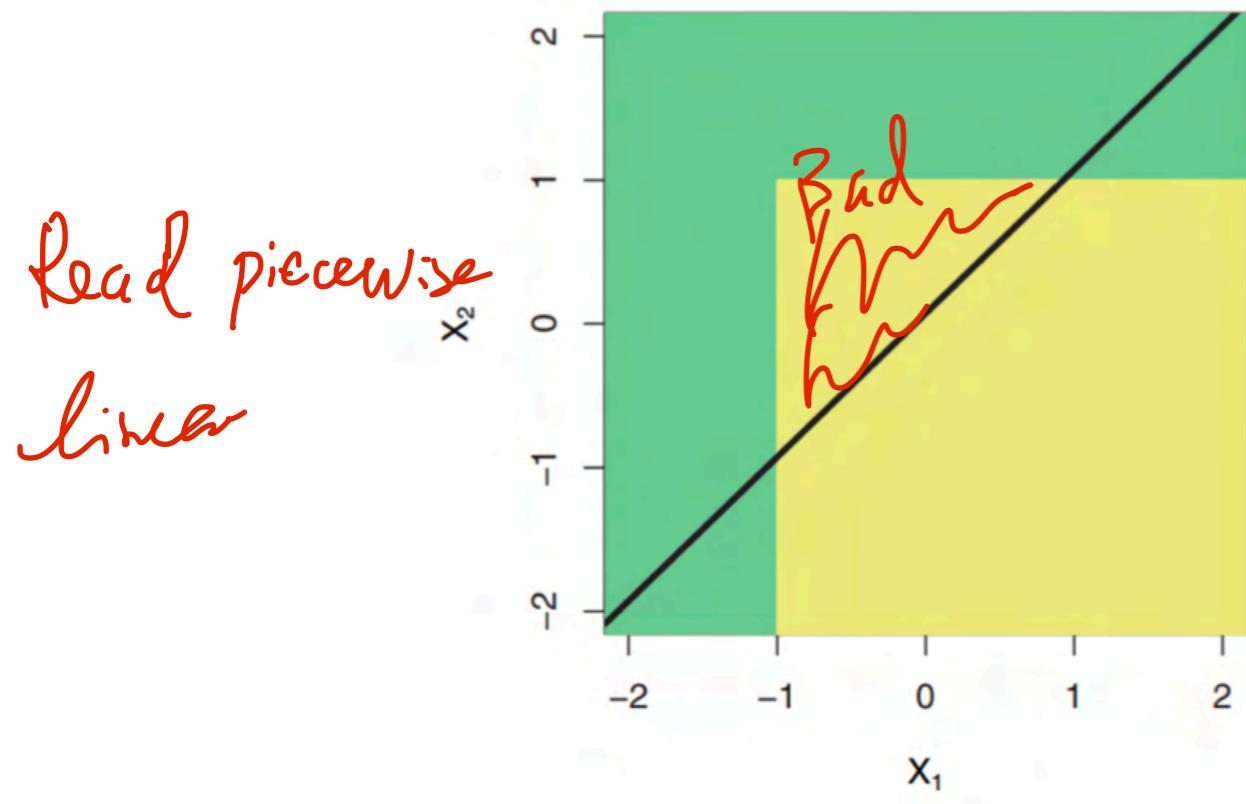
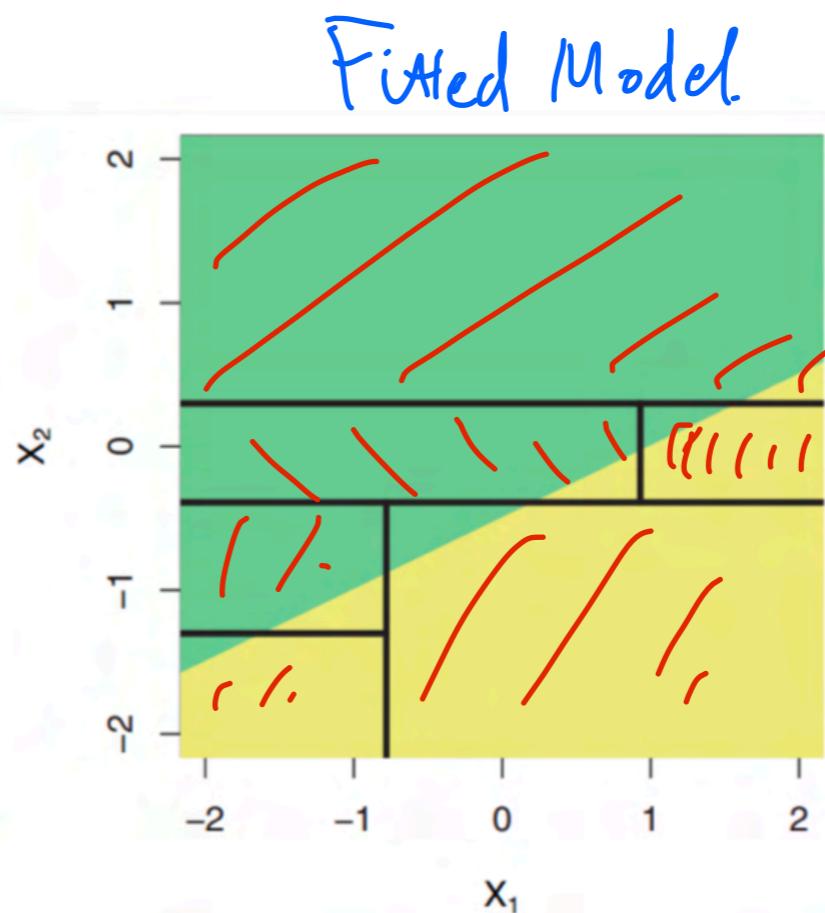
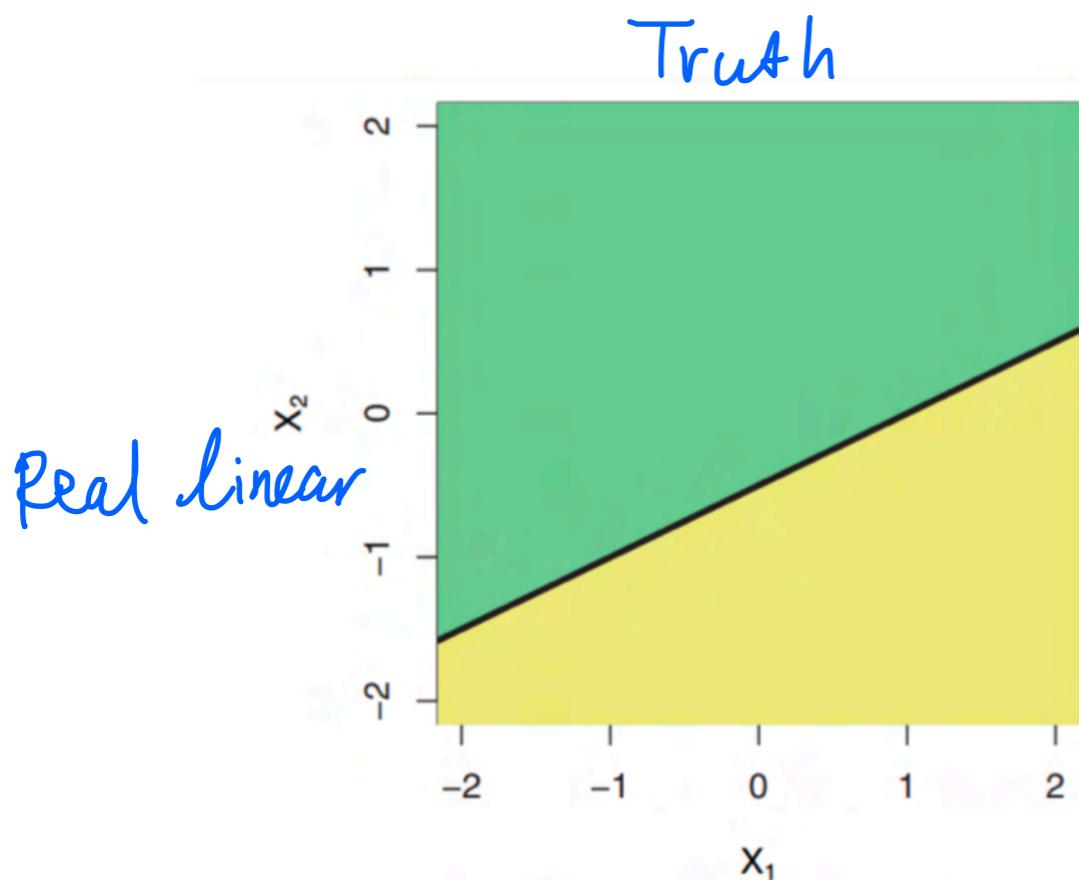
Work better in practice

loss
↑
The thermodynamics \Rightarrow entropy



\hat{p}_{mk} : prop of correct

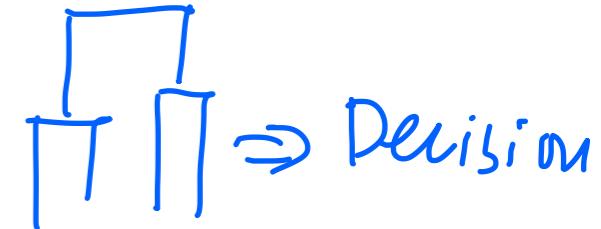
Tree versus Linear Models



Pros/Cons of CART

Pros

- Interpretable, and easy to explain to non-experts
- Fast to train and fast for prediction → Greedy
- Handle high dimensional data gracefully → Greedy : One variable each time
- Handles collinear/dependent predictors with no problems

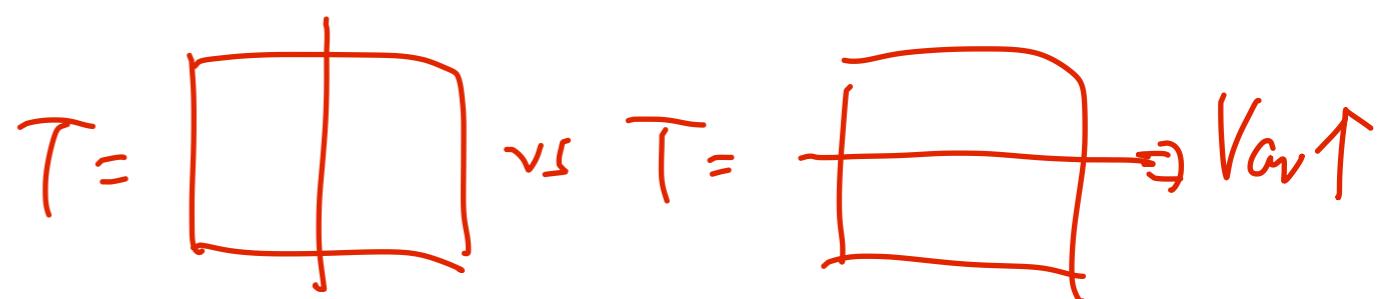


Cons

- Poor bias-variance trade-off
- Lack of smoothness

Piecewise constant is
not smooth.

: Large variance for tree : Sensitive
to greedy method



Bagging and Random Forest

Junwei Lu



HARVARD
T.H. CHAN

SCHOOL OF PUBLIC HEALTH
Department of Biostatistics



Pros/Cons of CART

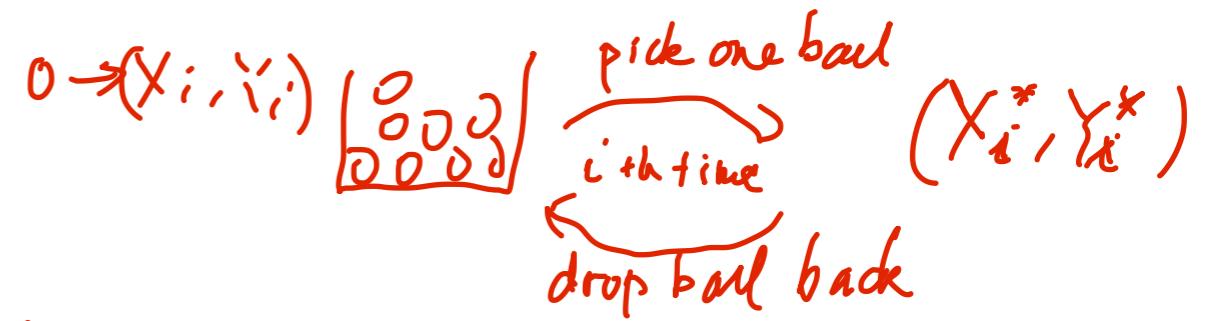
Pros

- Interpretable, and easy to explain to non-experts
- Fast to train and fast for prediction
- Handle high dimensional data gracefully
- Handles collinear/dependent predictors with no problems

Cons

- Poor bias-variance trade-off: High variance
- Lack of smoothness

Bagging to Reduce Variance



Main idea:

$$(Y_1, X_1), (Y_2, X_2), \dots, (Y_n, X_n)$$

Original data: Z_1, \dots, Z_n , where $Z_i = (Y_i, X_i)$

Bootstrap:

$$\begin{aligned} (X_1^{*1}, Y_1^{*1}) &= \\ (X_n^{*1}, Y_n^{*1}) &= \end{aligned}$$

$$Z_{1:n}^{*1}$$

Sample with replacement

$$Z_{1:n}^{*2}$$

$$\dots$$

$$Z_{1:n}^{*B-1}$$

$$Z_{1:n}^{*B}$$

Repeat for B times.

$$(X_1^{*B}, Y_1^{*B}) \dots$$

$$(X_n^{*B}, Y_n^{*B})$$

$$\hat{f}_1$$

$$\hat{f}_2$$

\dots

$$\hat{f}_{B-1}$$

$$\hat{f}_B$$

Tree

Tree

Tree

Tree

Tree

Reg: Bagging estimator $\Rightarrow \hat{f}^{\text{Bagging}}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^b(x)$ Arg results \Rightarrow Variance ↓

(Majority vote for classification)
k-NN

Bagging

$$\hat{f}^{\text{Bagging}}(x) \approx \frac{1}{B} \sum_{b=1}^B \hat{f}_b(x)$$

The variance of $\hat{f}^{\text{Bagging}}(x)$ should be smaller than single $\hat{f}_b(x) \rightarrow \text{High Variance}$

B doesn't need to be carefully tuned

$$z_{1:n}^1 \dots z_{1:n}^B \rightarrow \text{Bl Varf}$$

compt

Bagging is particularly useful for trees. We do not need prune.

$\overbrace{\downarrow}$
Avoid overfitting
Bagging avoid overfitting

Random Forest \Rightarrow Tree + Bagging + difference
Previous slide

Algorithm:

For $b = 1$ to B

Bagging

(a) Draw a bootstrap sample $Z_{1:n}^{*b}$ from $Z_{1:n}$

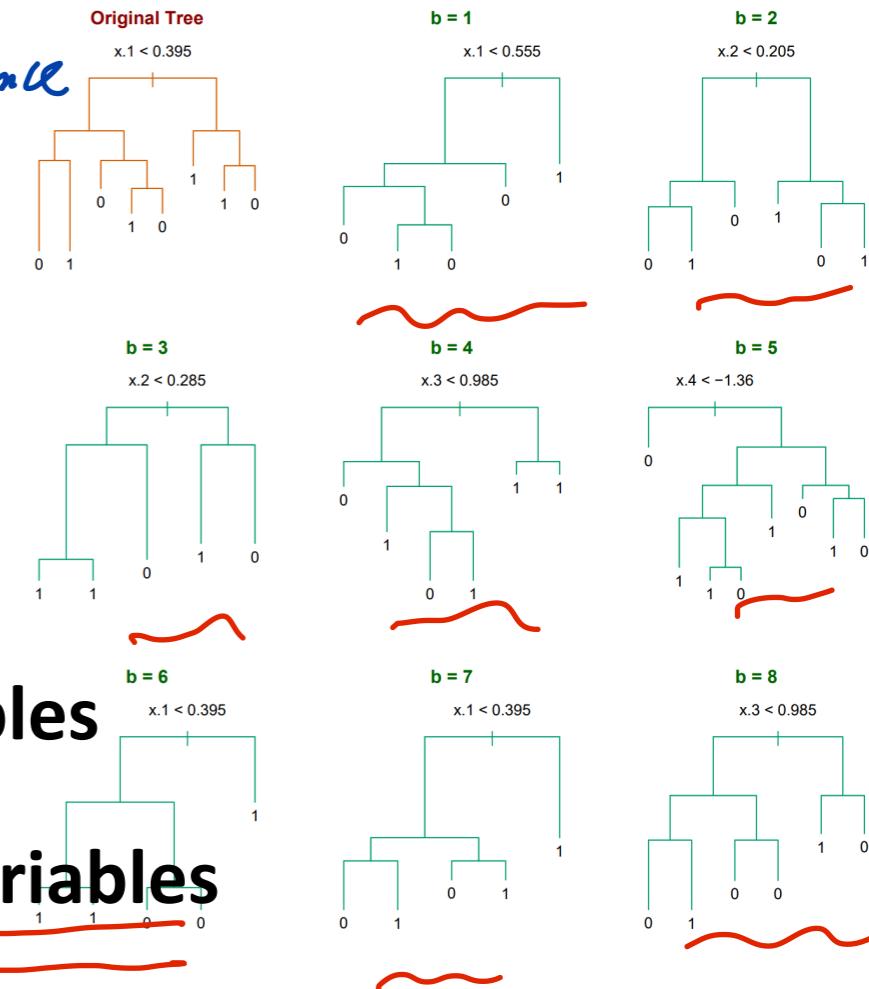
Dif(btree)

\Rightarrow (b) Select m variables at random from d variables

$m \leq d$

(c) Fit the tree $\hat{f}^b(x)$ using $Z_{1:n}^{*b}$ and these m variables

Output $\hat{f}^{\text{RF}}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^b(x)$



Compared to the naive bagging, random forest

focus on m -variables

m -variable

- forces the different trees to use different sources of information
- further reduces the variance \Rightarrow # Variables $m \leq d$

Random Forest



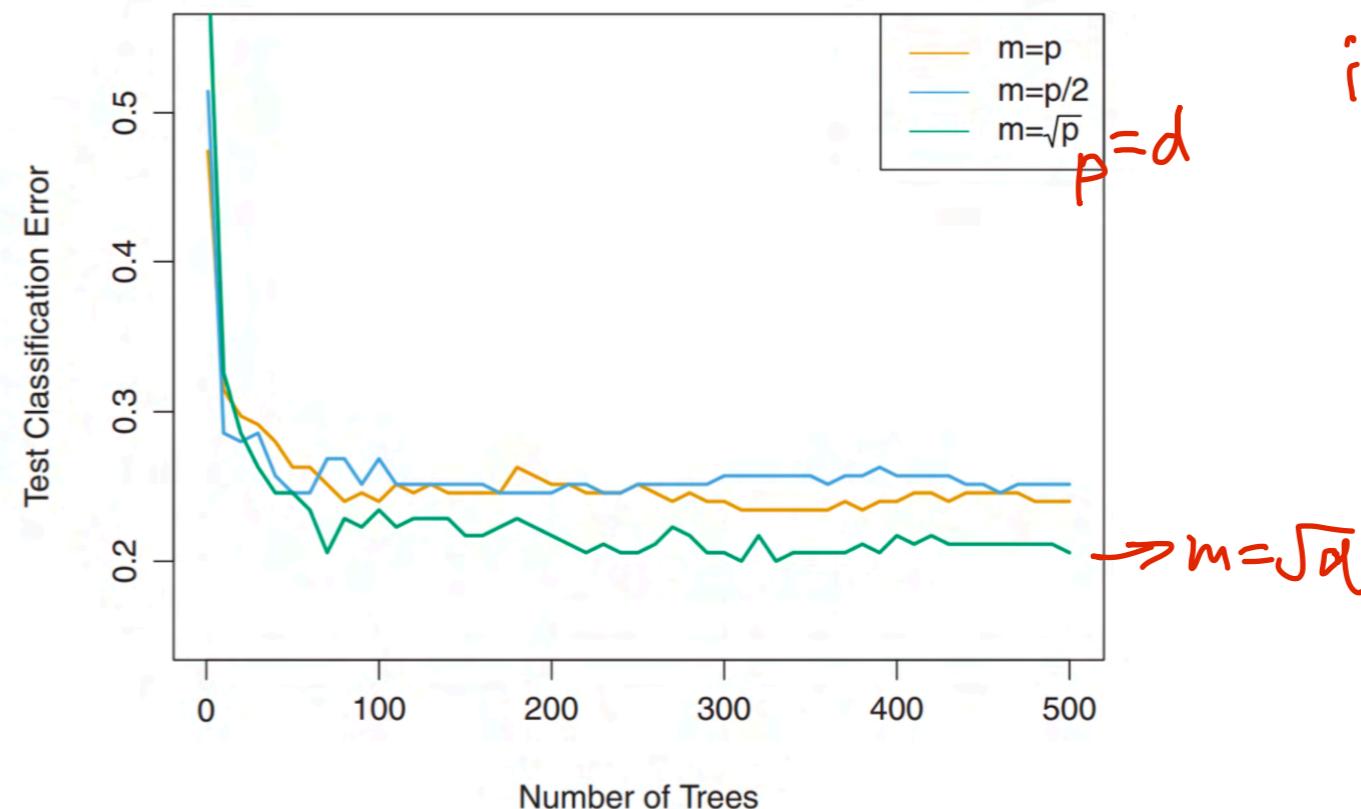
Question: How to choose m in practice?

Regression: $m \approx \underline{d/3}$

Classification: $m \approx \underline{\sqrt{d}}$

Choose smaller m when predictors are correlated

\Rightarrow *M forces you to look into different information*



Out of Bag (OOB) Error



Question: How to estimate the testing error \Rightarrow Tune parameter

Cross-validation : k -fold cross-risk

↓
too slow in computation

$(X_1, Y_1) \dots (X_n, Y_n)$

$\boxed{C_1 C_2 C_3}$

Given i , many bootstrap samples do not contain (Y_i, X_i)

\downarrow
 (X_i, Y_i)

\Downarrow Bootstrap

We denote their index set to be $C^{-i} \subseteq \{1, \dots, B\}$

$b=1$ $b=2$

$N_b(X_i, Y_i)$

$\boxed{C_1 C_2 C_3}$ $b=1$
 $\leftarrow \boxed{C_2 C_3 C_1}$ $b=2$

\vdots
 $\boxed{C_1 C_2 C_3}$ $b=B$

Out of Bag Error (Leave-one-out prediction error) is

$$\text{Test err} = \hat{\text{MSE}}_{\text{OOB}} = \frac{1}{n} \sum_{i=1}^n \frac{1}{|C^{-i}|} \sum_{b \in C^{-i}} \text{Test without } i (Y_i - \hat{f}^b(X_i))^2$$

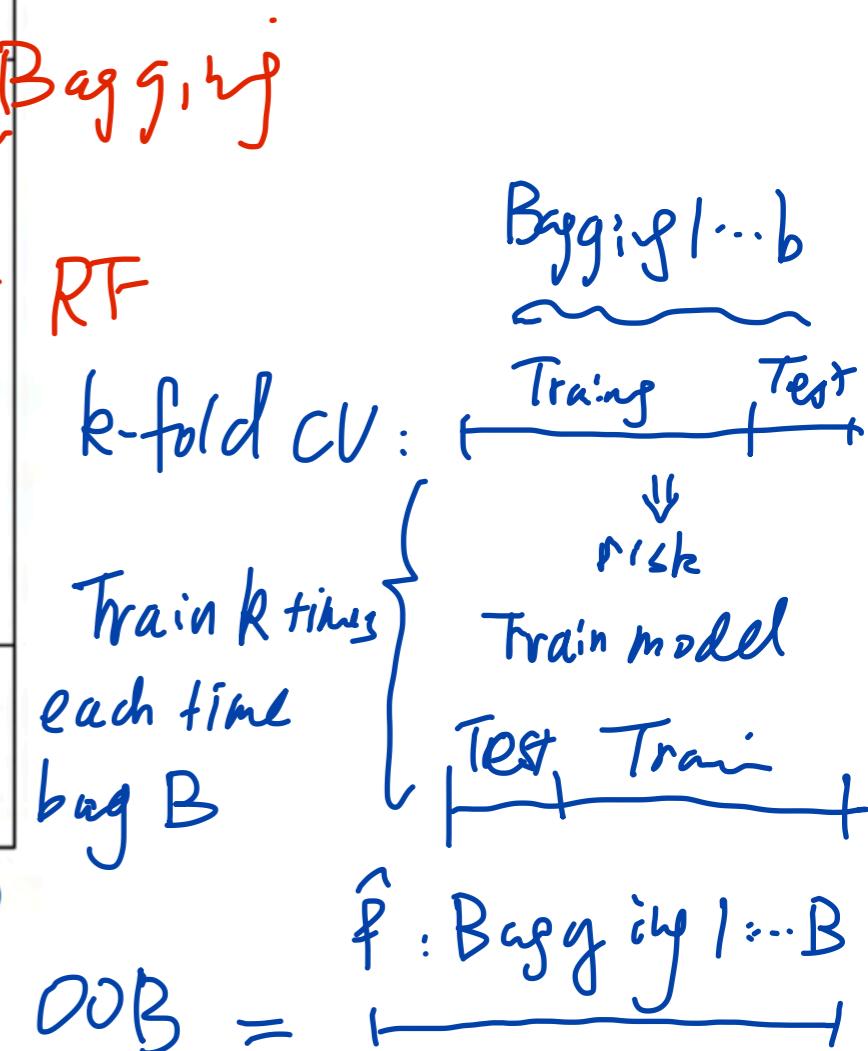
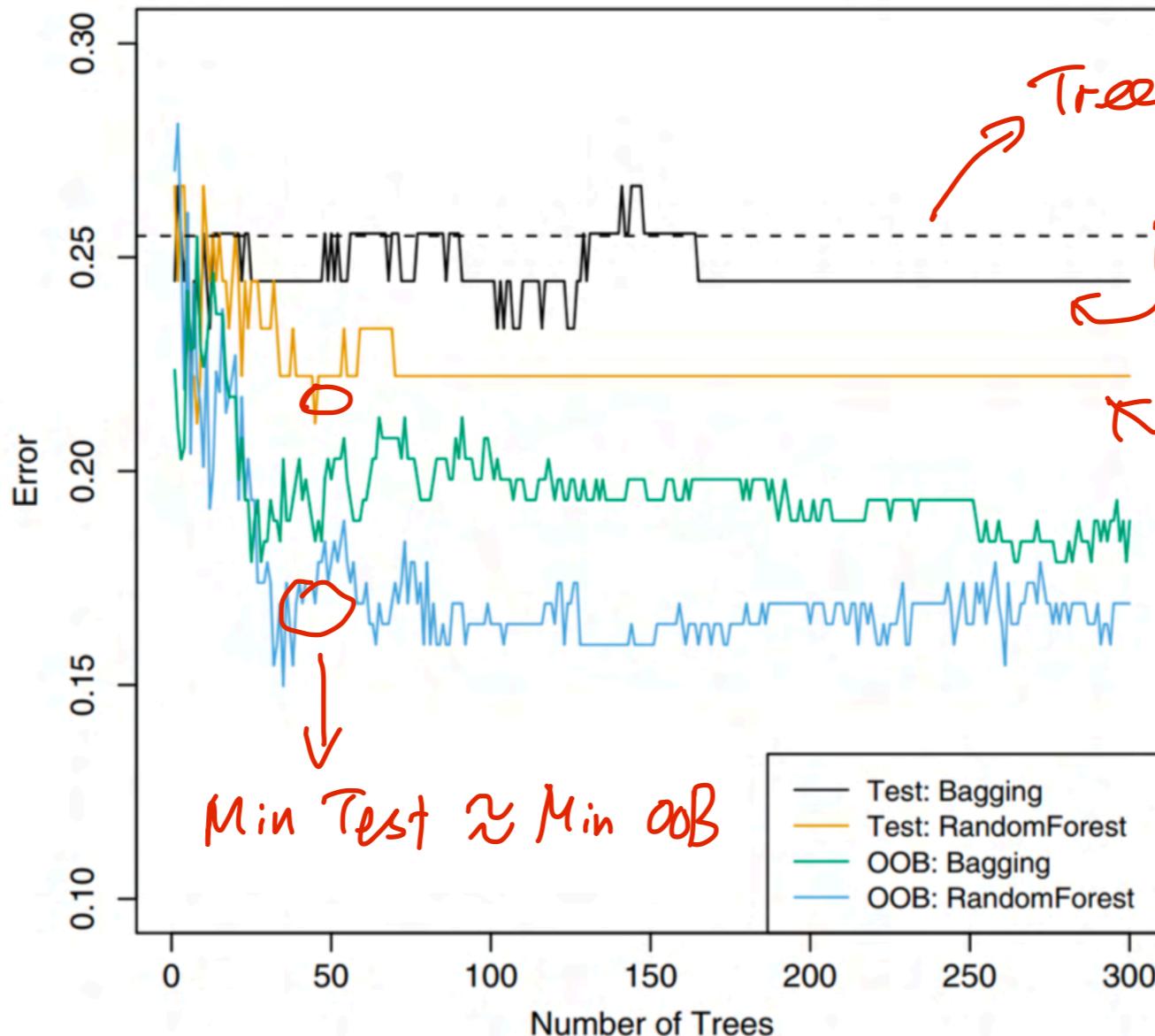
Pros of Out of Bag Error:

Faster than cross-validation

prediction err by leaving i out.

Out of Bag (OOB) Error

Bagging only improve slightly over a single tree (dashed line)



Pros/Cons of Random Forest

Pros

- Often, better performance due to lower variance. \Rightarrow Bagging
- OOB error estimate is convenient. \Rightarrow Faster than CV
- Less variance and computation than bagging.
 $m \text{ variables} \leq d$

Cons

- More computation than single trees \Rightarrow Bagging fit tree for B times
- Lower interpretability than single trees.

$$\frac{1}{B} \left(T_1 + T_2 + T_3 \right) = RF$$

good to understand

Bagging and Random Forest

Junwei Lu



HARVARD
T.H. CHAN

SCHOOL OF PUBLIC HEALTH
Department of Biostatistics



Pros/Cons of CART

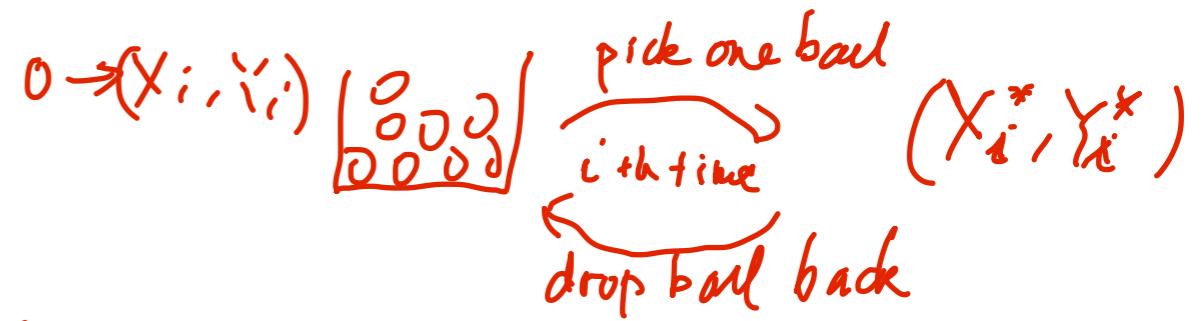
Pros

- Interpretable, and easy to explain to non-experts
- Fast to train and fast for prediction
- Handle high dimensional data gracefully
- Handles collinear/dependent predictors with no problems

Cons

- Poor bias-variance trade-off: High variance
- Lack of smoothness

Bagging to Reduce Variance



Main idea:

$$(Y_1, X_1), (Y_2, X_2), \dots, (Y_n, X_n)$$

Original data: Z_1, \dots, Z_n , where $Z_i = (Y_i, X_i)$

Bootstrap:

$$\begin{aligned} (X_1^{*1}, Y_1^{*1}) &= \\ (X_n^{*1}, Y_n^{*1}) &= \end{aligned}$$

$$Z_{1:n}^{*1}$$

Sample with replacement

$$Z_{1:n}^{*2}$$

$$\dots$$

$$Z_{1:n}^{*B-1}$$

$$Z_{1:n}^{*B}$$

Repeat for B times.

$$(X_1^{*B}, Y_1^{*B}) \dots$$

$$(X_n^{*B}, Y_n^{*B})$$

$$\hat{f}_1$$

$$\hat{f}_2$$

\dots

$$\hat{f}_{B-1}$$

$$\hat{f}_B$$

Tree

Tree

Tree

Tree

Tree

Reg: Bagging estimator $\Rightarrow \hat{f}^{\text{Bagging}}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^b(x)$ Arg results \Rightarrow Variance ↓

(Majority vote for classification)
k-NN

Bagging

$$\hat{f}^{\text{Bagging}}(x) \approx \frac{1}{B} \sum_{b=1}^B \hat{f}_b(x)$$

The variance of $\hat{f}^{\text{Bagging}}(x)$ should be smaller than single $\hat{f}_b(x) \rightarrow \text{High Variance}$

B doesn't need to be carefully tuned

$$z_{1:n}^1 \dots z_{1:n}^B \rightarrow \text{Bl Varf}$$

compt

Bagging is particularly useful for trees. We do not need prune.

$\overbrace{\downarrow}$
Avoid overfitting
Bagging avoid overfitting

Random Forest \Rightarrow Tree + Bagging + difference
Previous slide

Algorithm:

For $b = 1$ to B

Bagging

(a) Draw a bootstrap sample $Z_{1:n}^{*b}$ from $Z_{1:n}$

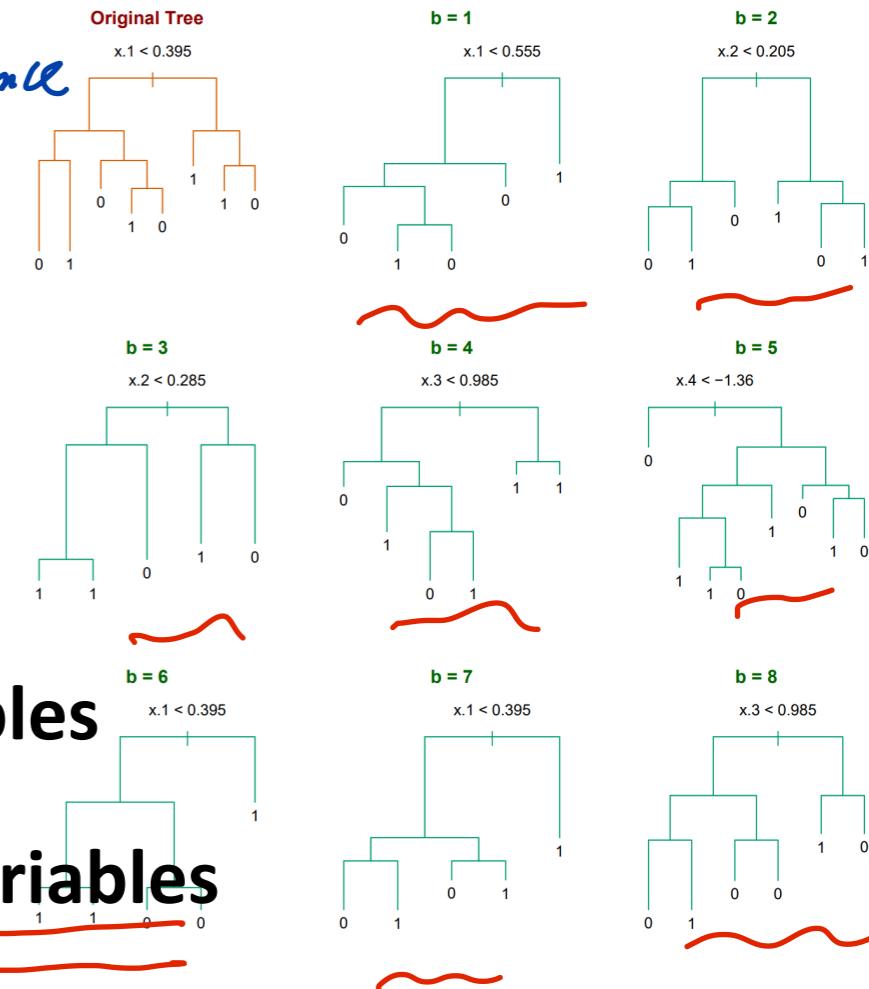
Dif(btree)

\Rightarrow (b) Select m variables at random from d variables

$m \leq d$

(c) Fit the tree $\hat{f}^b(x)$ using $Z_{1:n}^{*b}$ and these m variables

Output $\hat{f}^{\text{RF}}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^b(x)$



Compared to the naive bagging, random forest

focus on m -variables

m -variable

- forces the different trees to use different sources of information
- further reduces the variance \Rightarrow # Variables $m \leq d$

Random Forest



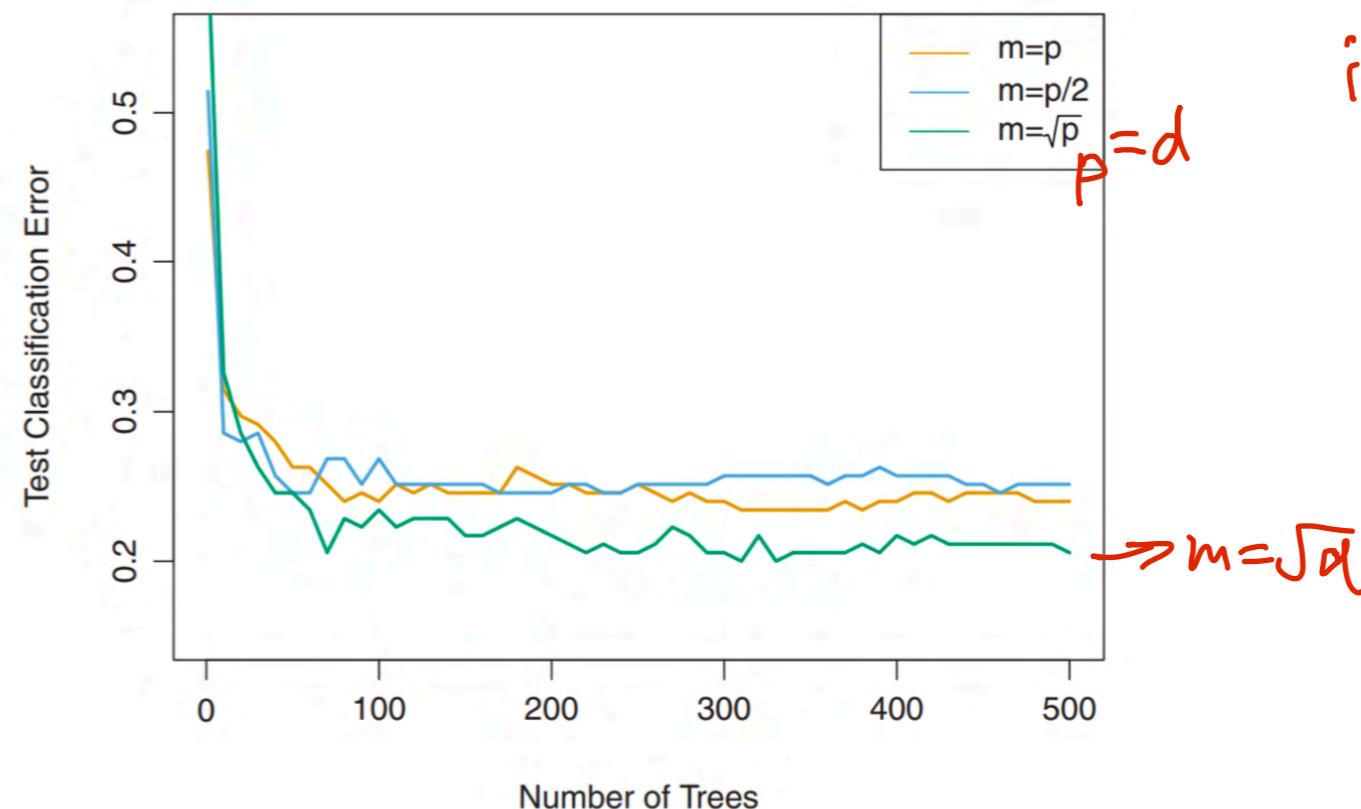
Question: How to choose m in practice?

Regression: $m \approx \underline{d/3}$

Classification: $m \approx \underline{\sqrt{d}}$

Choose smaller m when predictors are correlated

\Rightarrow *M forces you to look into different information*



Out of Bag (OOB) Error



Question: How to estimate the testing error \Rightarrow Tune parameter

Cross-validation : k -fold cross-risk

↓
too slow in computation

$(X_1, Y_1) \dots (X_n, Y_n)$

$\boxed{1 2 3}$

Given i , many bootstrap samples do not contain (Y_i, X_i)

\downarrow
 (X_i, Y_i)

\Downarrow Bootstrap

We denote their index set to be $C^{-i} \subseteq \{1, \dots, B\}$

$b=1$ $b=2$

$N_b(X_i, Y_i)$

$\boxed{1 2}$

$b=1$
 $\boxed{1 2 3}$

$b=2$
 $\boxed{1 2 3}$

$b=3$
 $\boxed{1 2 3}$

Out of Bag Error (Leave-one-out prediction error) is

$$\text{Test err} = \hat{\text{MSE}}_{\text{OOB}} = \frac{1}{n} \sum_{i=1}^n \frac{1}{|C^{-i}|} \sum_{b \in C^{-i}} \left(Y_i - \hat{f}^b(X_i) \right)^2$$

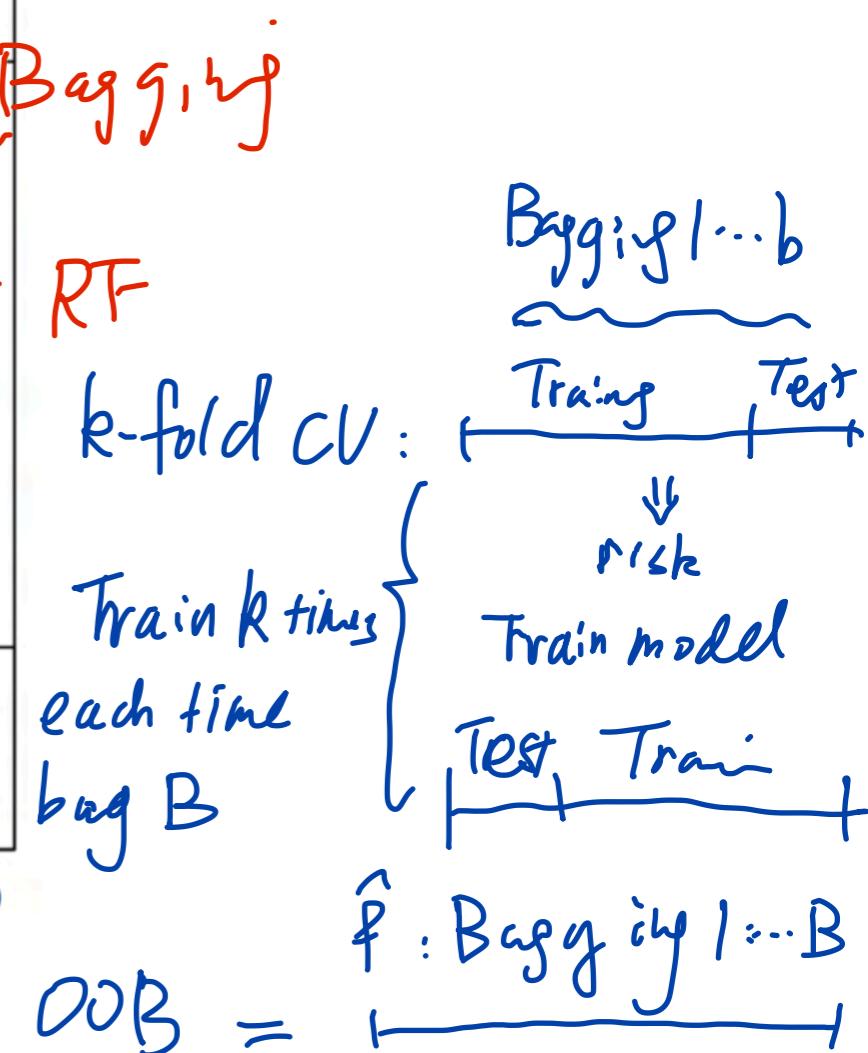
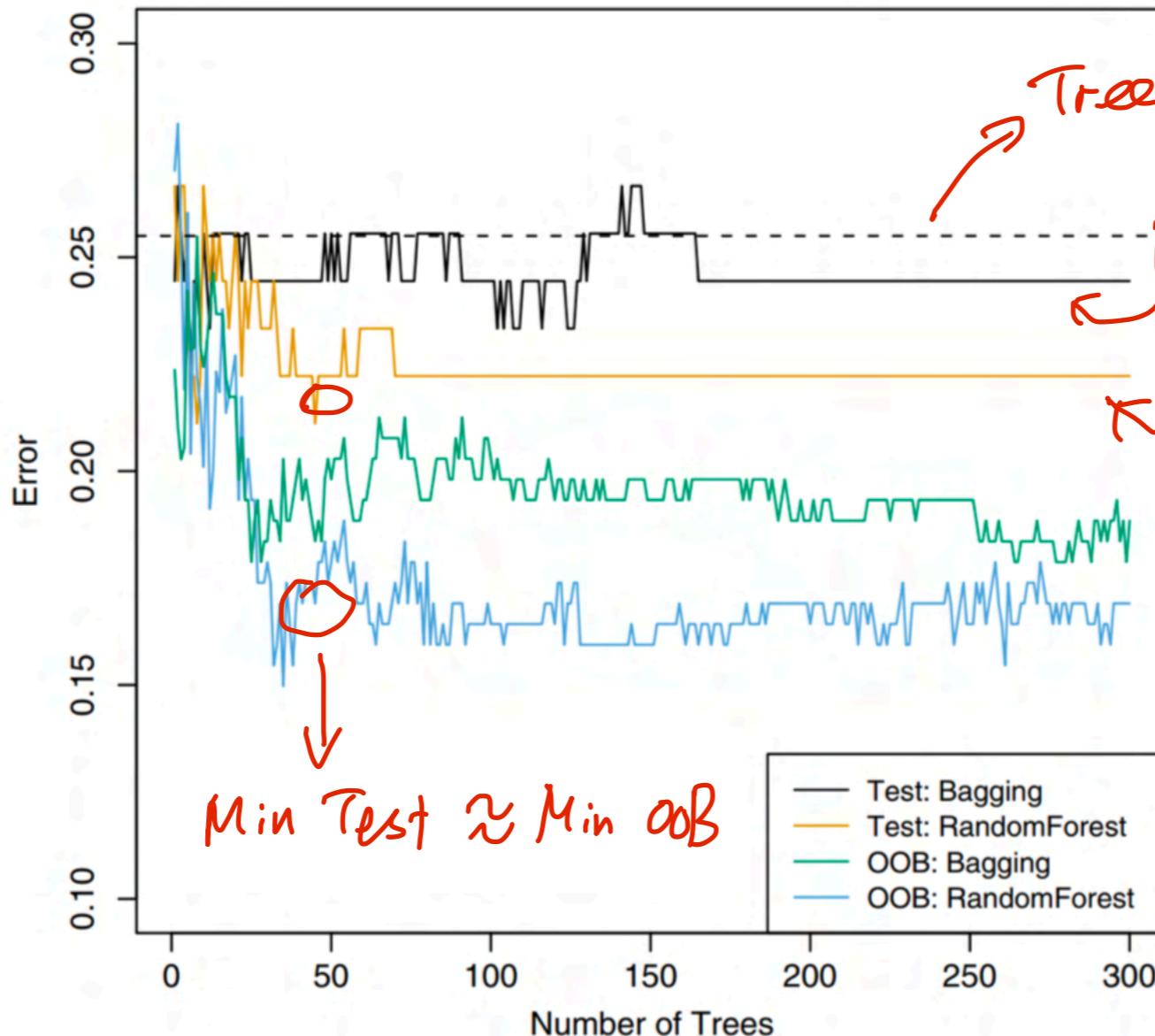
Pros of Out of Bag Error:

Faster than cross-validation

prediction err by leaving i out.

Out of Bag (OOB) Error

Bagging only improve slightly over a single tree (dashed line)



Pros/Cons of Random Forest

Pros

- Often, better performance due to lower variance. \Rightarrow Bagging
- OOB error estimate is convenient. \Rightarrow Faster than CV
- Less variance and computation than bagging.
 $m \text{ variables} \leq d$

Cons

- More computation than single trees \Rightarrow Bagging fit tree for B times
- Lower interpretability than single trees.

$$\frac{1}{B} \left(T_1 + T_2 + T_3 \right) = RF$$

good to understand

Deep Learning

Junwei Lu

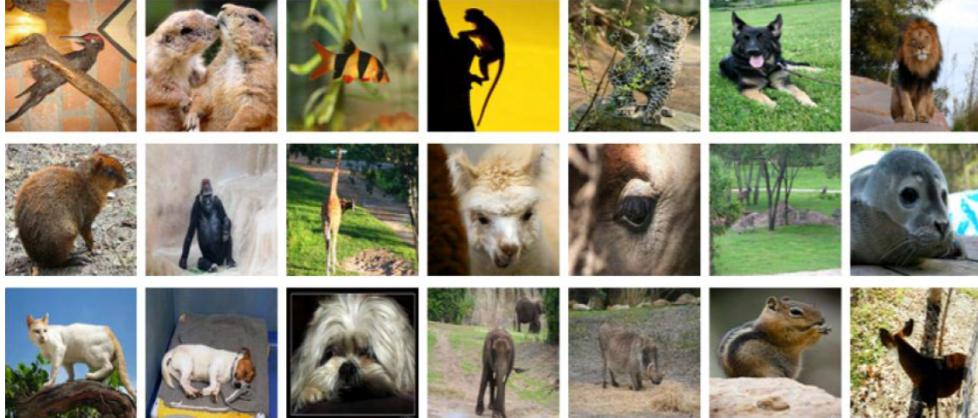


HARVARD
T.H. CHAN

SCHOOL OF PUBLIC HEALTH
Department of Biostatistics



Deep Learning: Applications



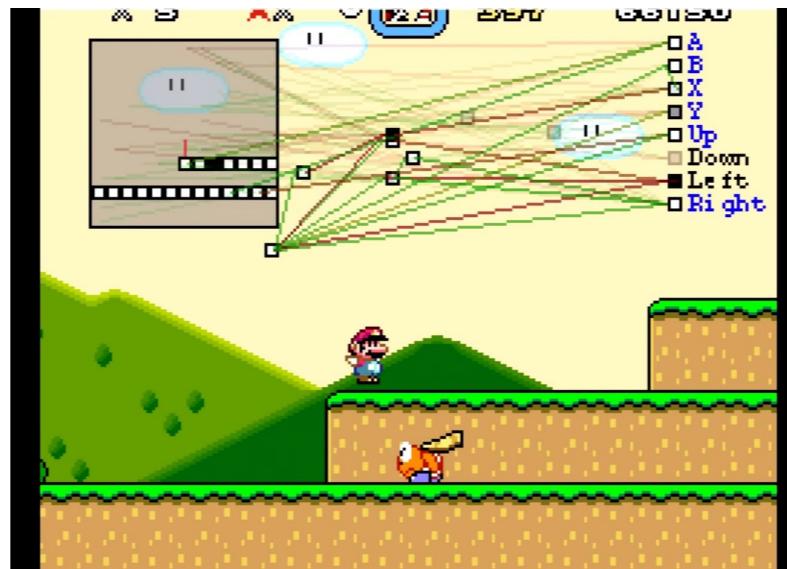
Computer vision



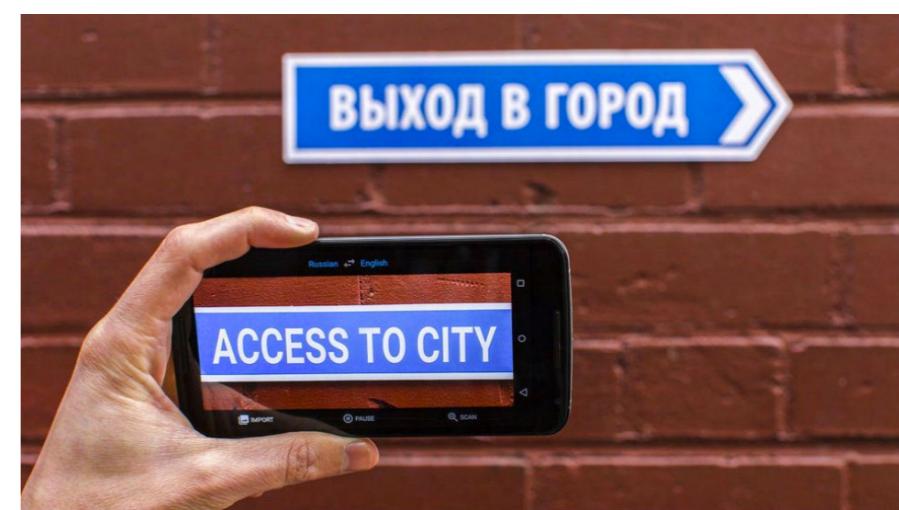
Self-driving car



AlphaGo



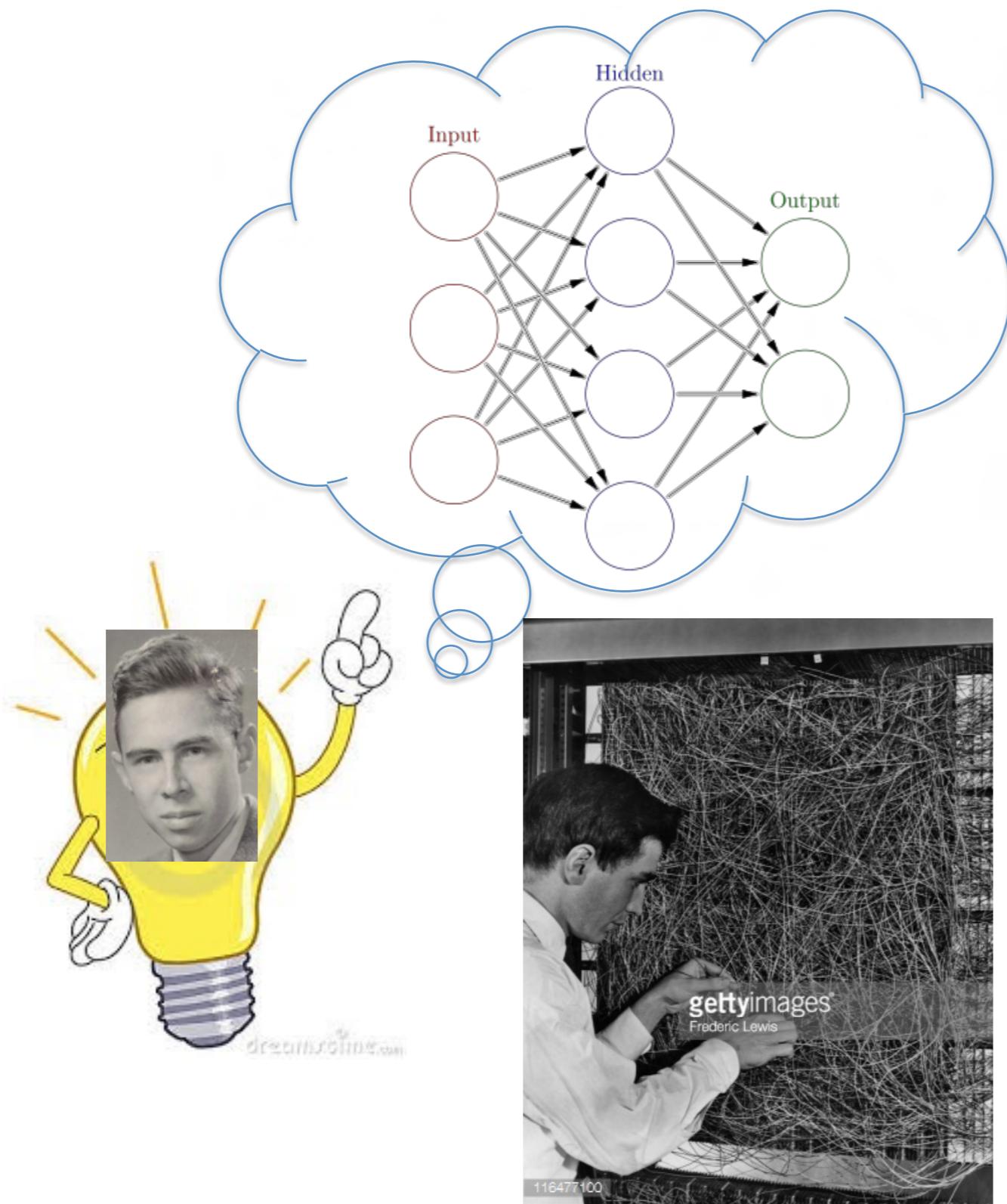
Super mario



Google Translate

**Deep learning is not invented
over night....**

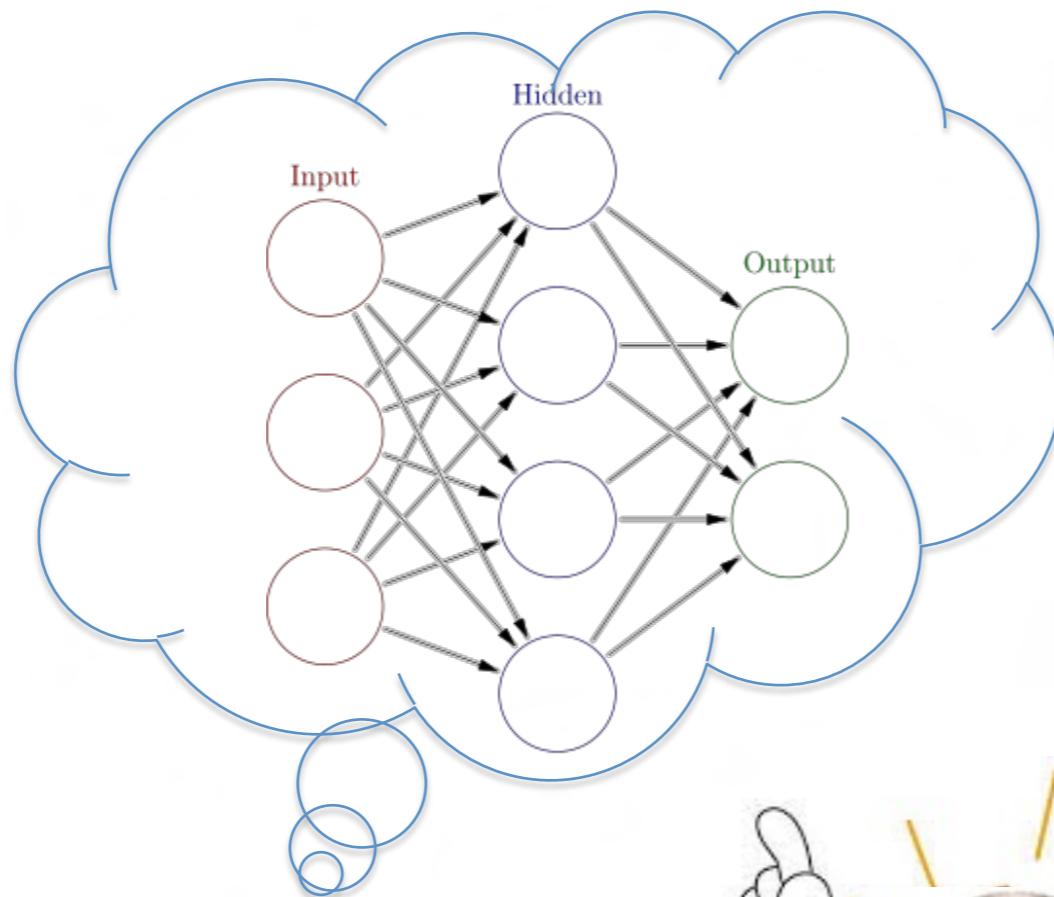
History of Deep Learning



1957
Frank Rosenblatt
Artificial neural network

Mark I Perceptron

History of Deep Learning

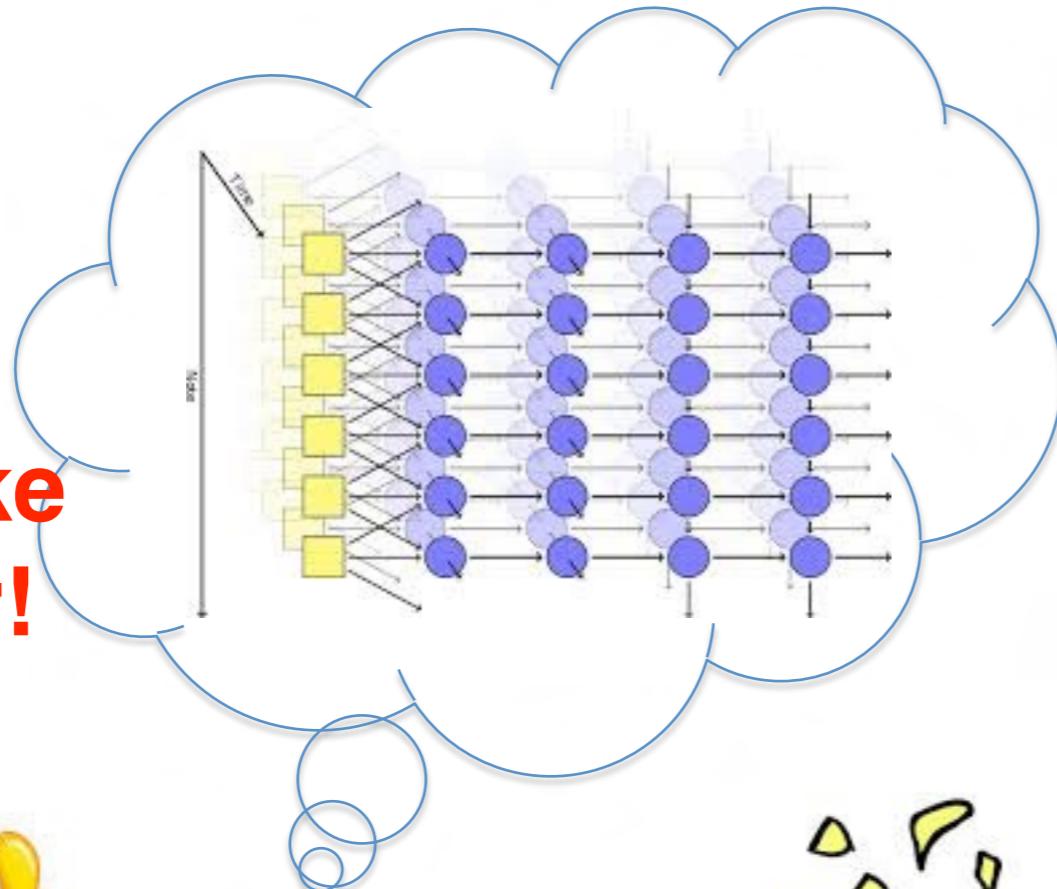


1957
Frank Rosenblatt
Artificial neural network

1993
Vladimir Vapnik
Support vector machine

History of Deep Learning

**Let's make
it deeper!**



- 1957
Frank Rosenblatt
**Artificial neural
network**
- 1993
Vladimir Vapnik
**Support vector
machine**
- 2006
Geoffrey Hinton ...
Deep neural network

Deep Learning Trinity

1. Architecture

Supervised

- Fully connected nn
- Convolutional nn
- Recurrent nn
- & thousands variations

Unsupervised

- Autoencoder
- Deep belief network
- Generative Adversarial Net
- & thousands variations

2. Training

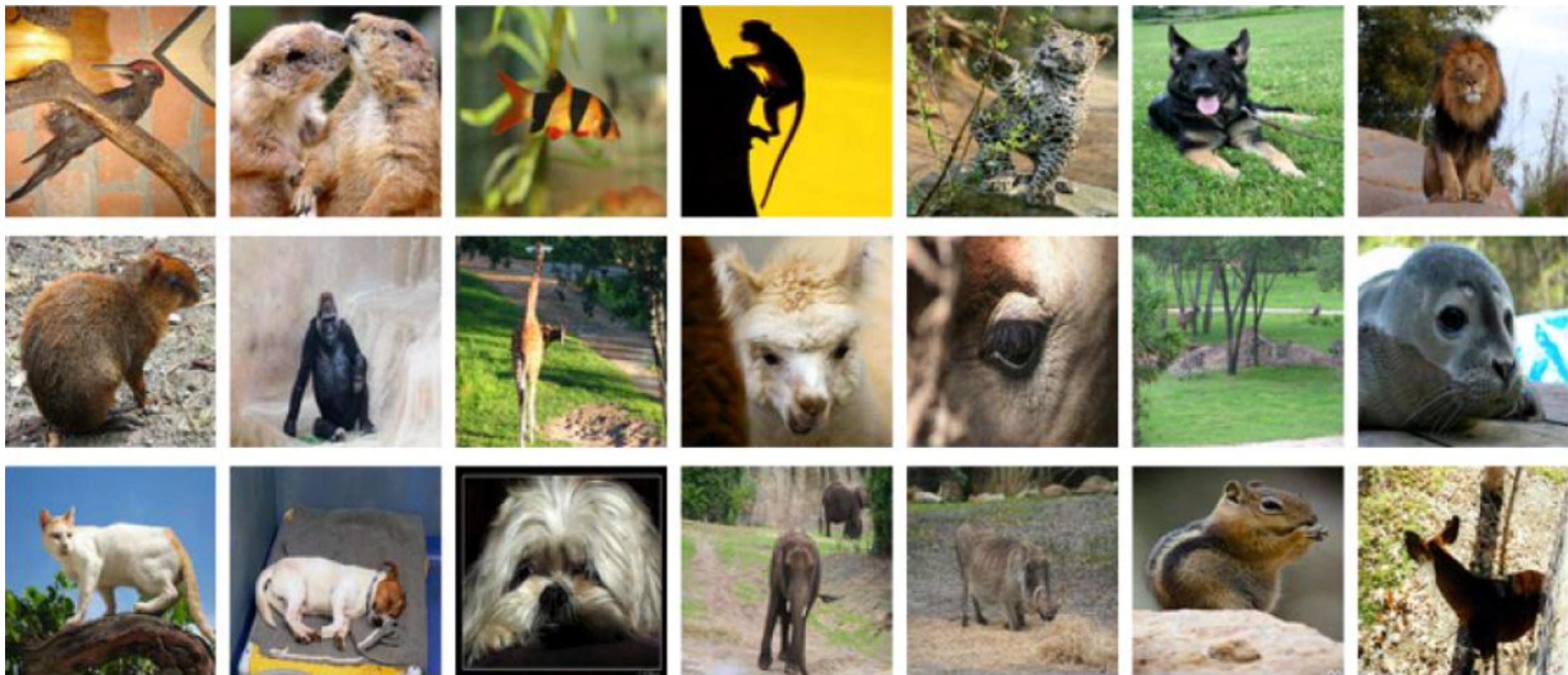
- Stochastic gradient descent
- Dropout
- Pretraining
- & thousands variations

3. Applications

- Many dirty works...

CNN: ImageNet Dataset

- 1K categories
- 1.2M training images
- 150,000 testing images



ImageNet Dataset Result



mite

container ship

motor scooter

leopard

mite	container ship	motor scooter	leopard
black widow cockroach tick starfish	lifeboat amphibian fireboat drilling platform	go-kart moped bumper car golfcart	jaguar cheetah snow leopard Egyptian cat



grille

mushroom

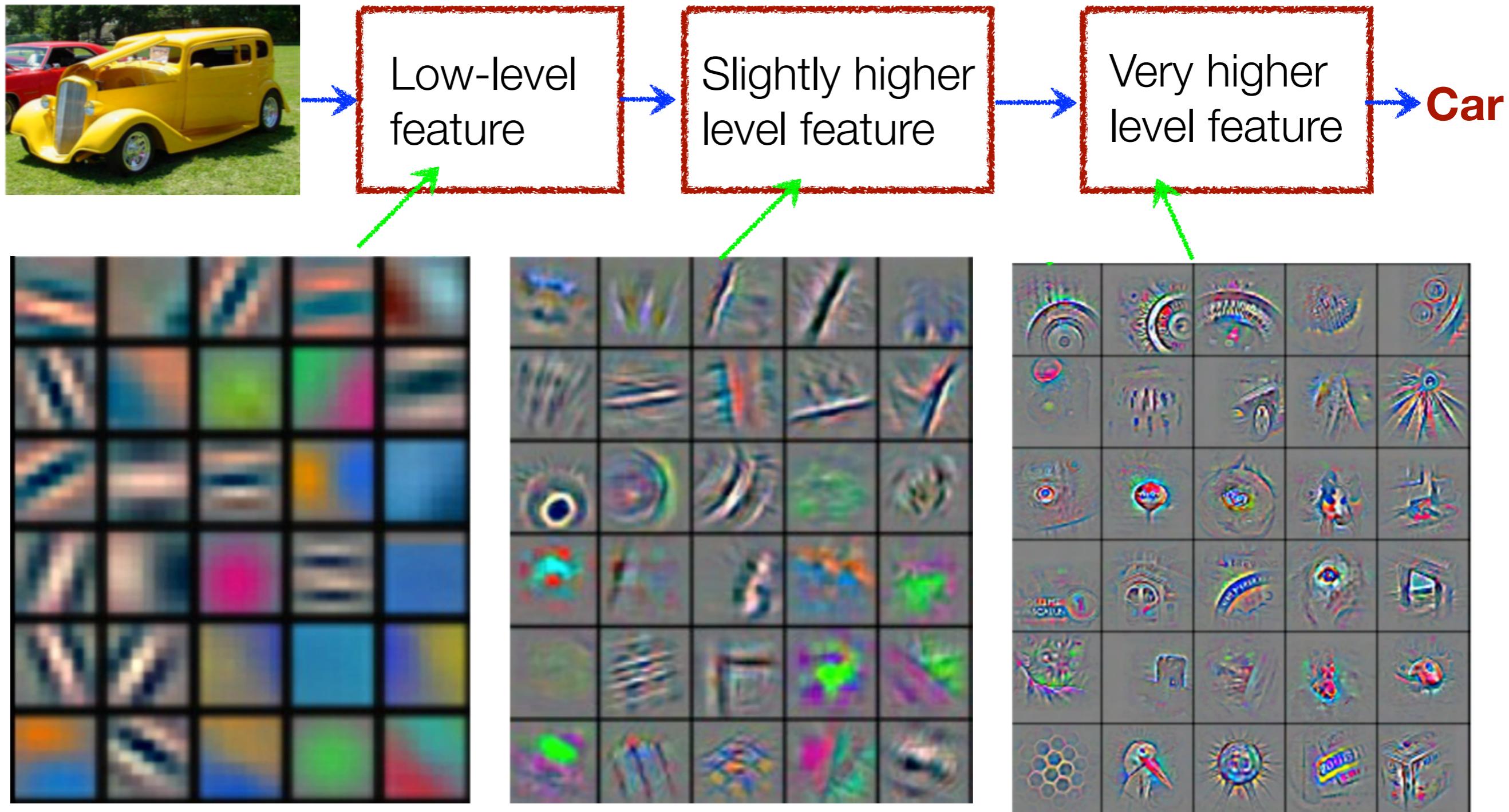
cherry

Madagascar cat

convertible grille pickup beach wagon fire engine	agaric mushroom jelly fungus gill fungus dead-man's-fingers	dalmatian grape elderberry ffordshire bullterrier currant	squirrel monkey spider monkey titi indri howler monkey
---	---	---	--

Intuition of CNN

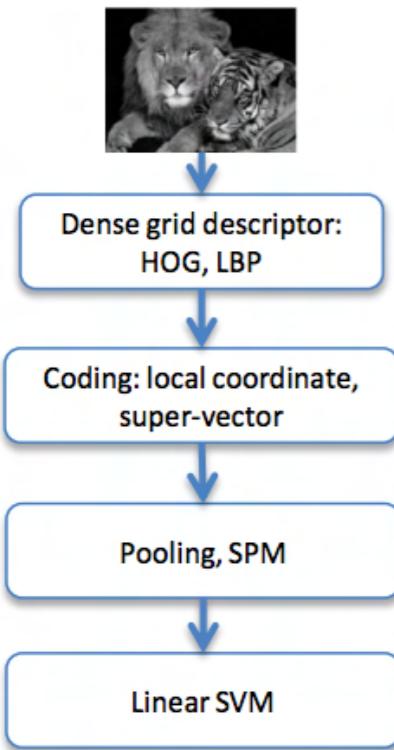
- Deep learning aims at learning **feature hierarchies**



ImageNet Dataset Result

Year 2010

NEC-UIUC

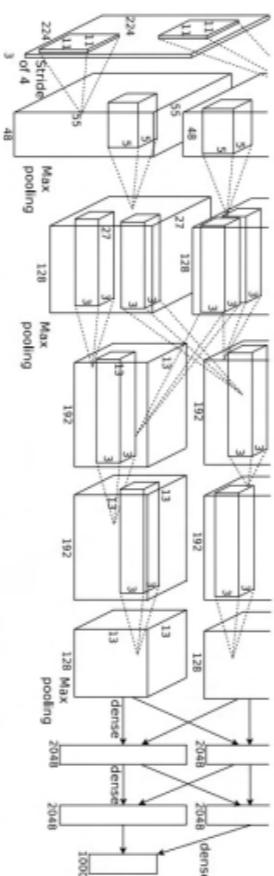


[Lin CVPR 2011]

25%

Year 2012

SuperVision



[Krizhevsky NIPS 2012]

20%

Year 2014

GoogLeNet

VGG

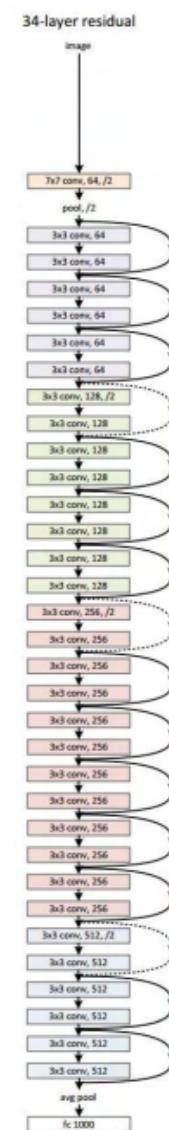


[Szegedy arxiv 2014]

16%

Year 2015

MSRA



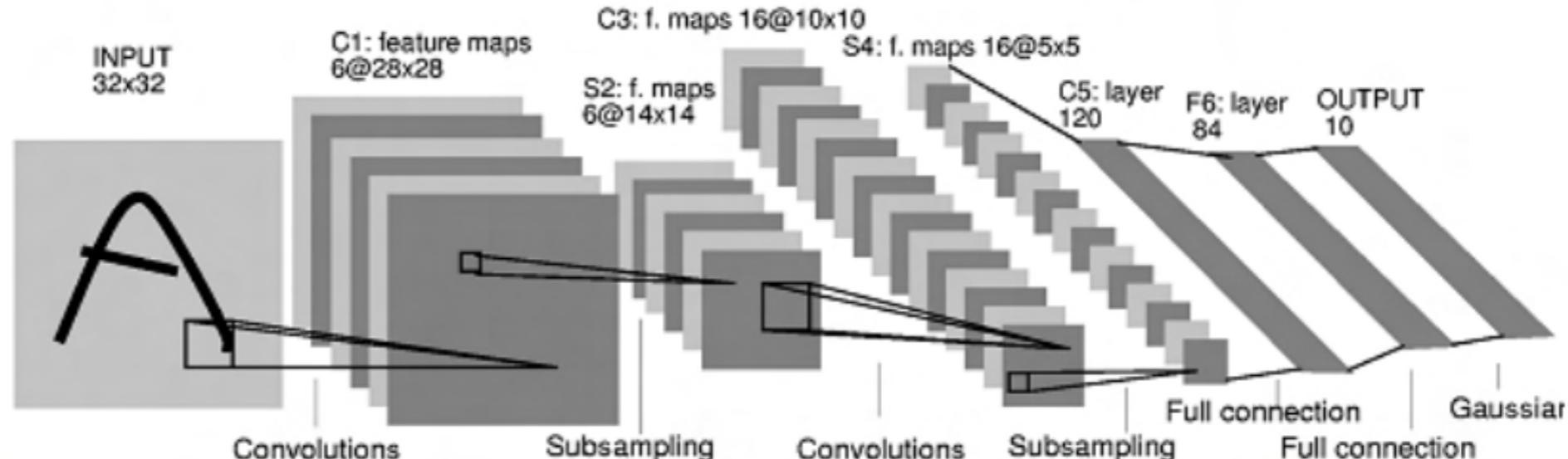
[Simonyan arxiv 2014]

3.5%

Better than human

1998

LeCun et al.



of transistors

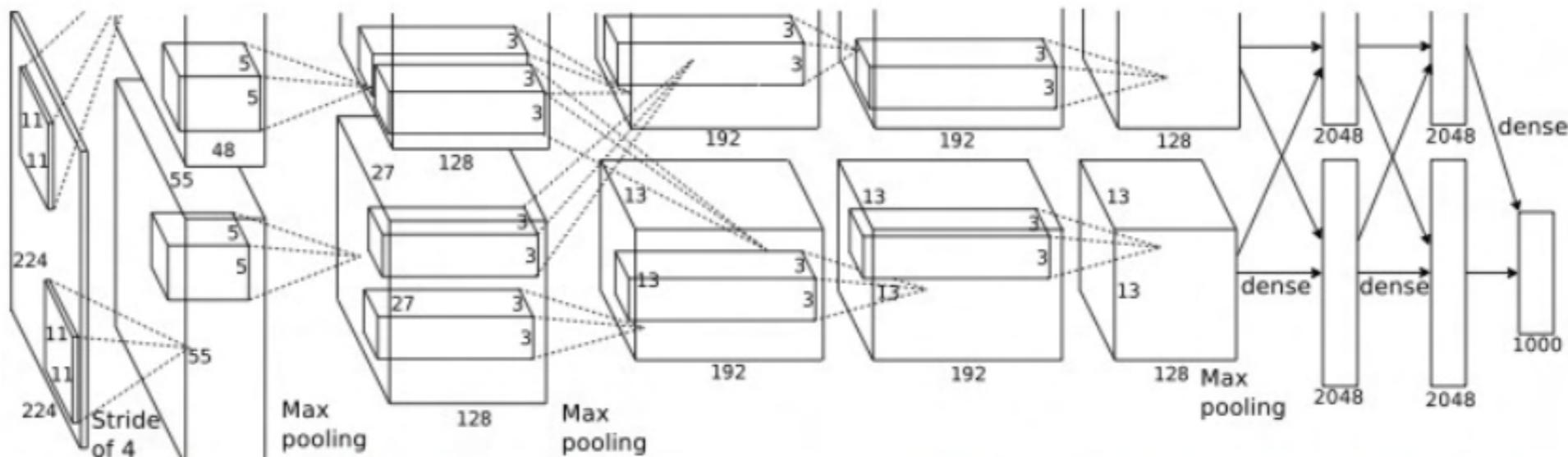


of pixels used in training

10^7 **NIST**

2012

Krizhevsky
et al.



of transistors GPUs



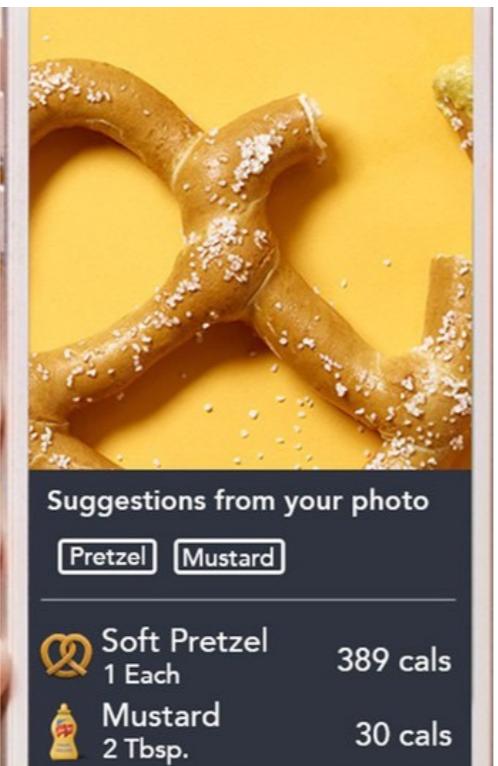
of pixels used in training



10^{14} **IMAGENET**

ImageNet Boosts Applications

Google Goggles



Neural style

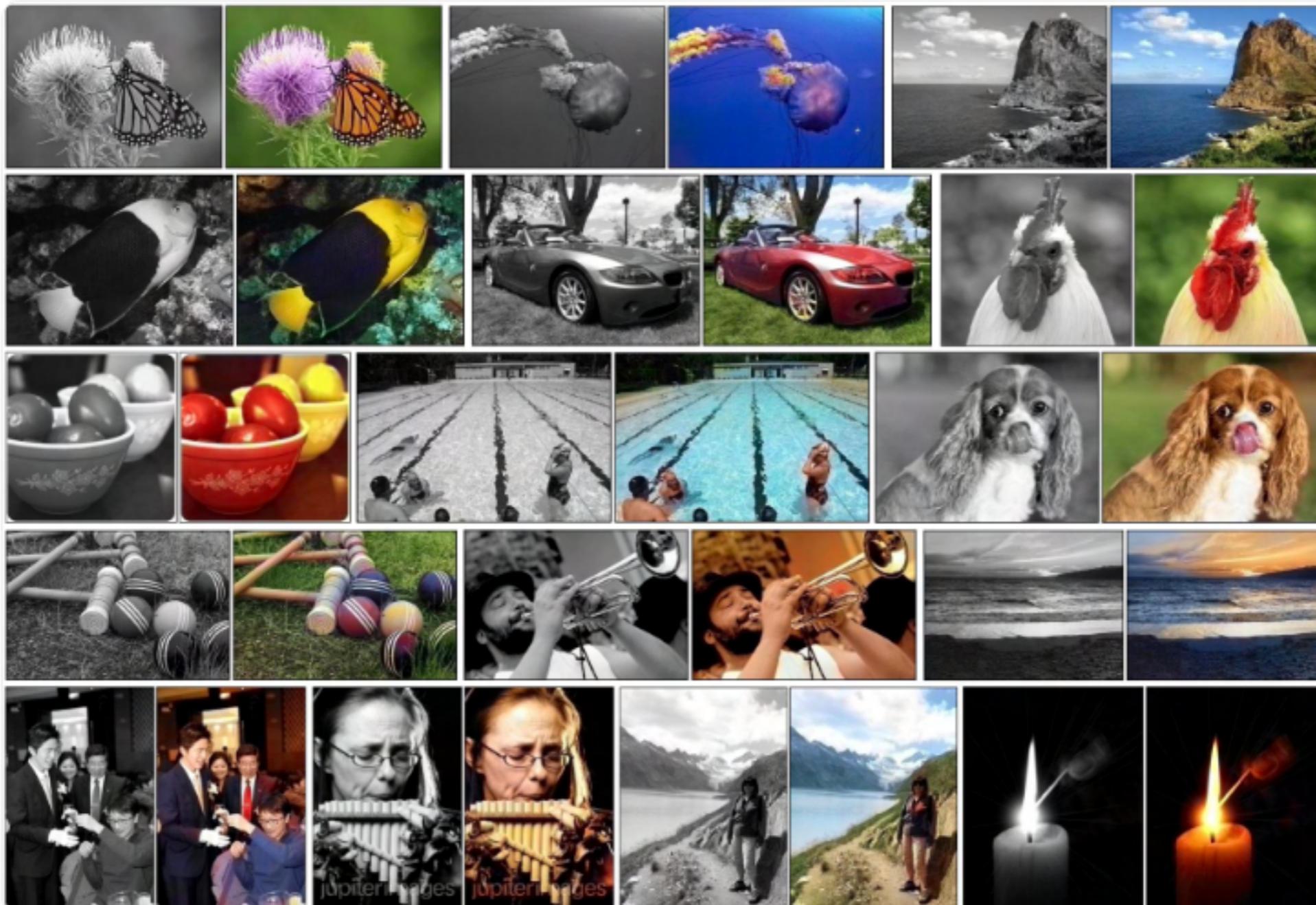


Deep dream



ImageNet Boosts Applications

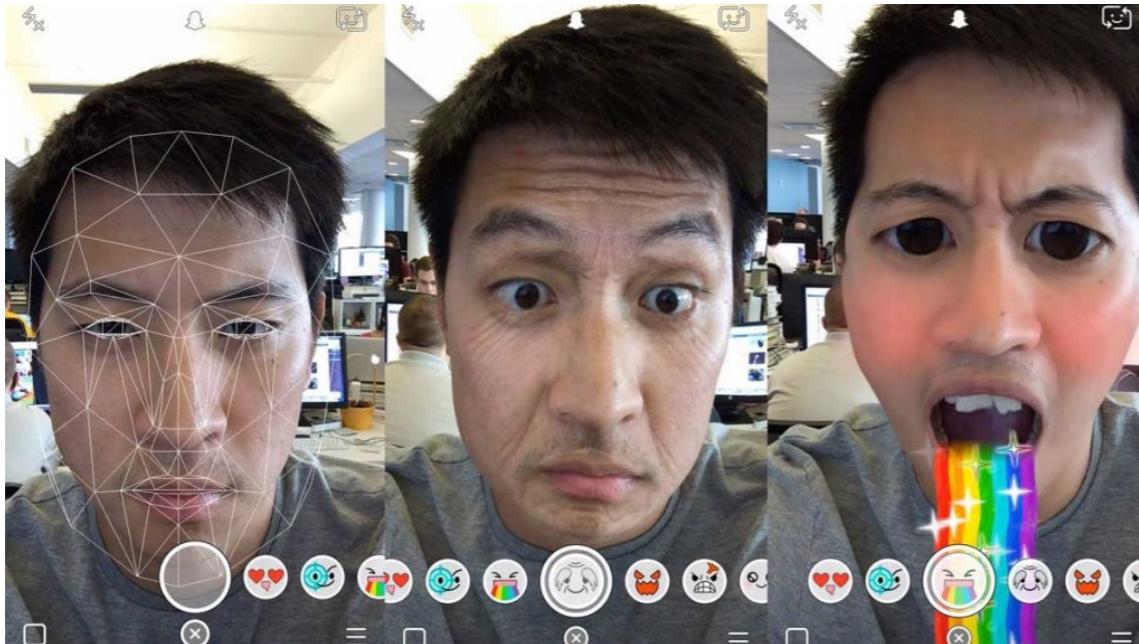
Automatic Colorization



<http://demos.algorithmia.com/colorize-photos/>

ImageNet Boosts Applications

Snapchat filters



Meitu app filters



PRISMA

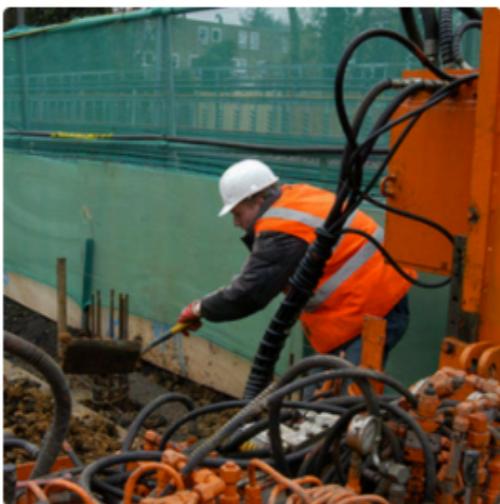


Recurrent NN: Natural Language Processing

Word2Vec: From images to languages Automated Image Captioning



"man in black shirt is playing guitar."



"construction worker in orange safety vest is working on road."



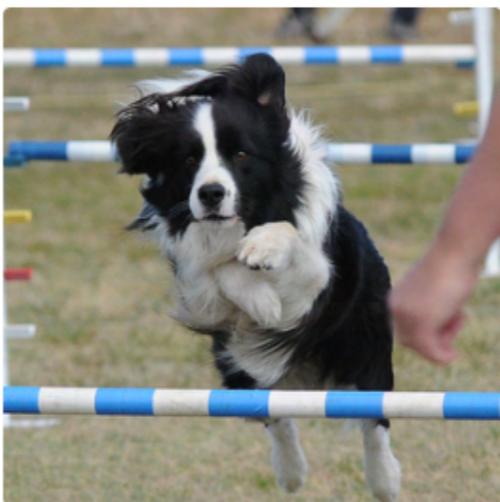
"two young girls are playing with lego toy."



"boy is doing backflip on wakeboard."



"girl in pink dress is jumping in air."



"black and white dog jumps over bar."



"young girl in pink shirt is swinging on swing."



"man in blue wetsuit is surfing on wave."

Recurrent NN: Natural Language Processing

Word2Vec: From images to languages
Automated Image Captioning



"a young boy is holding a baseball bat."



"a cat is sitting on a couch with a remote control."



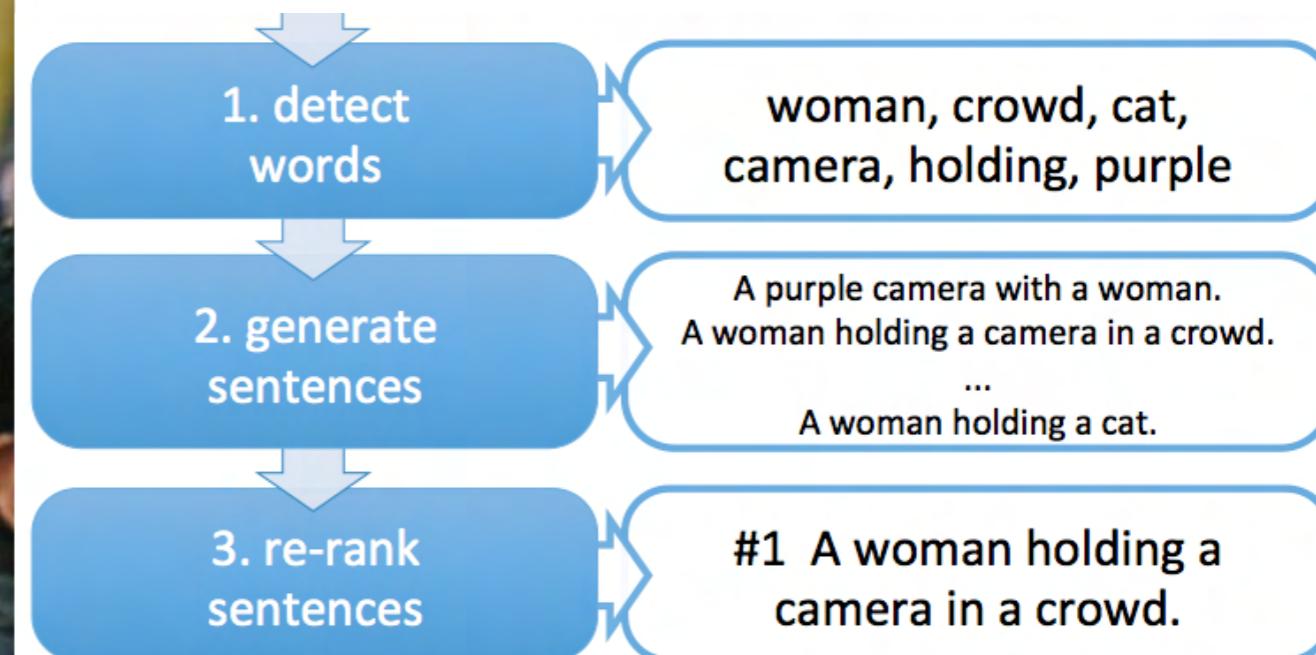
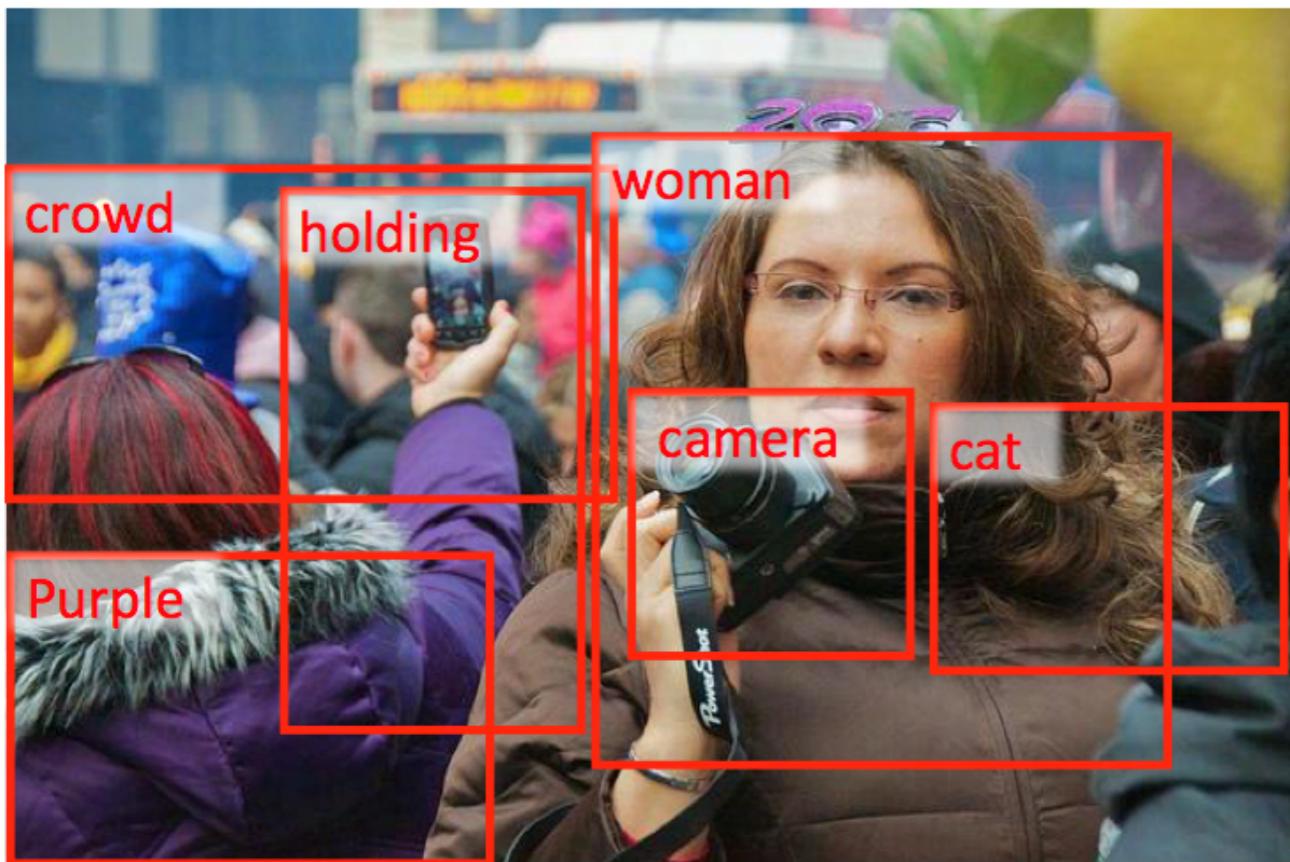
"a woman holding a teddy bear in front of a mirror."



"a horse is standing in the middle of a road."

Recurrent NN: Natural Language Processing

Word2Vec: From images to languages
Automated Image Captioning



Recurrent NN: Natural Language Processing

Word2Vec: From images to languages

Idea of word embedding

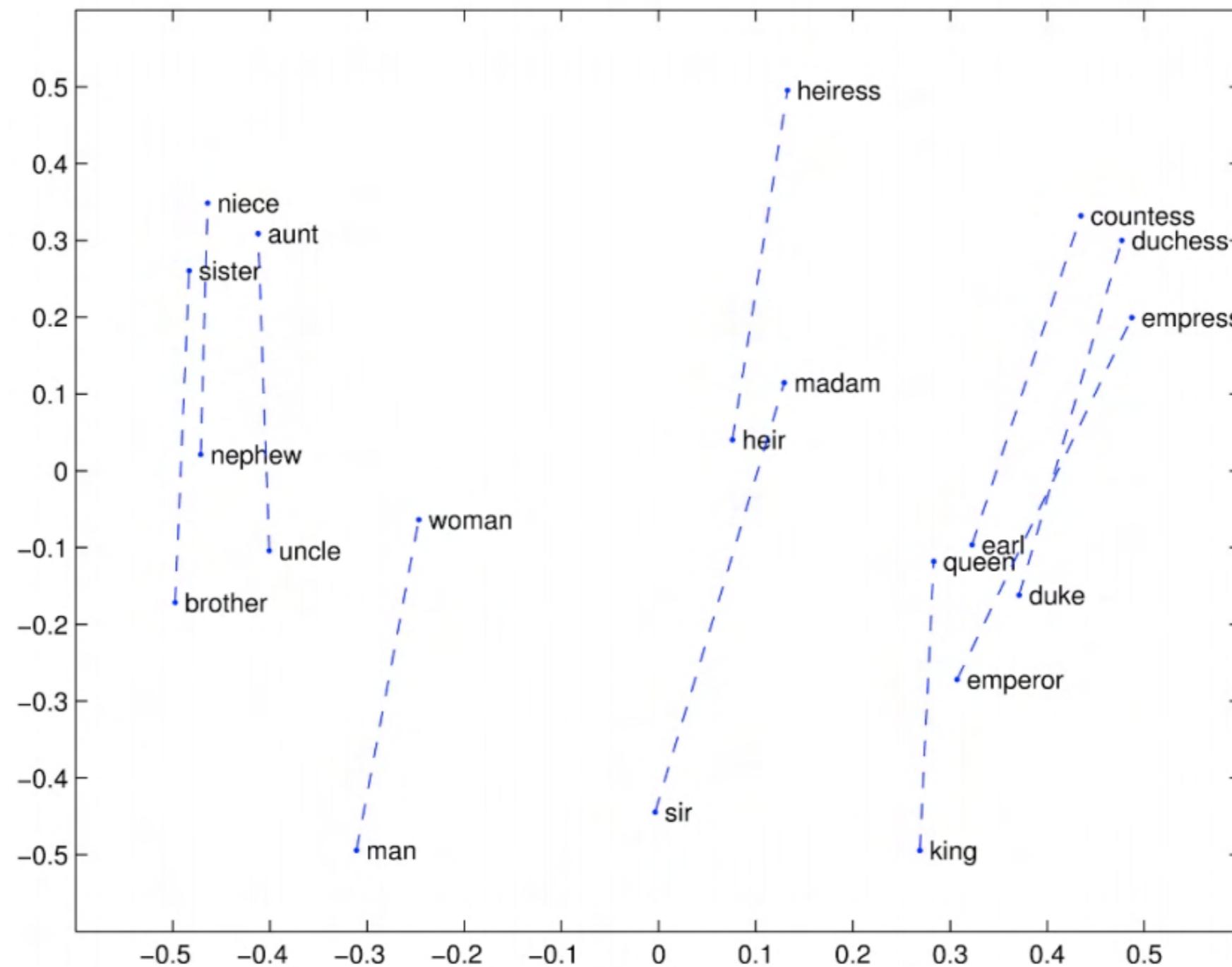
You shall know a word by the company it keeps!

government debt problems turning into banking crises as has happened in
saying that Europe needs unified banking regulation to replace the hodgepodge

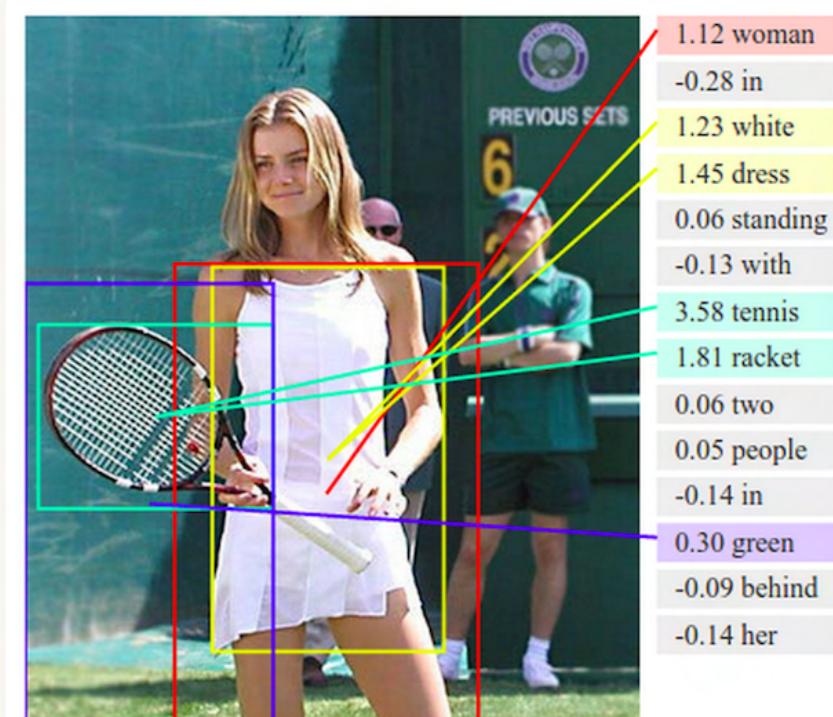
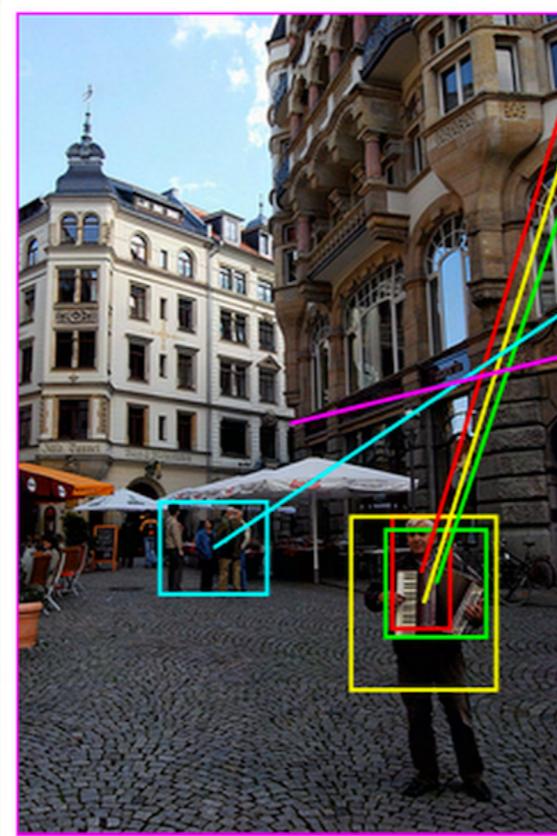
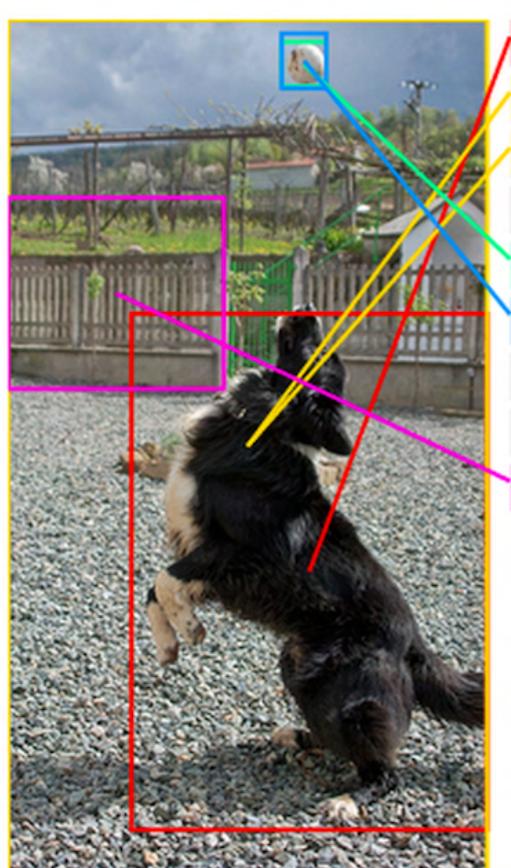
These words will represent **banking**

Recurrent NN: Natural Language Processing

Word2Vec: From images to languages
Idea of word embedding



Recurrent NN: Natural Language Processing



Recurrent NN: Applications



Google Translate



Automatic Handwriting Generation

We should build the wall and make mexico pay for it! Donald Trump

We should build the wall and make mexico pay for it! Donald Trump

We should build the wall and make mexico pay for it! Donald Trump

Recurrent NN: Applications

Youtube auto caption



Deep Chopin

Chopin Nocturne 1

Chopin Nocturne 2

Fake algebraic geometry paper

For $\bigoplus_{n=1,\dots,m} \mathcal{L}_{m_n} = 0$, hence we can find a closed subset \mathcal{H} in \mathcal{H} and any sets \mathcal{F} on X , U is a closed immersion of S , then $U \rightarrow T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \text{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \rightarrow V$. Consider the maps M along the set of points Sch_{fppf} and $U \rightarrow U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ???. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\text{Spec}(R') \rightarrow S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

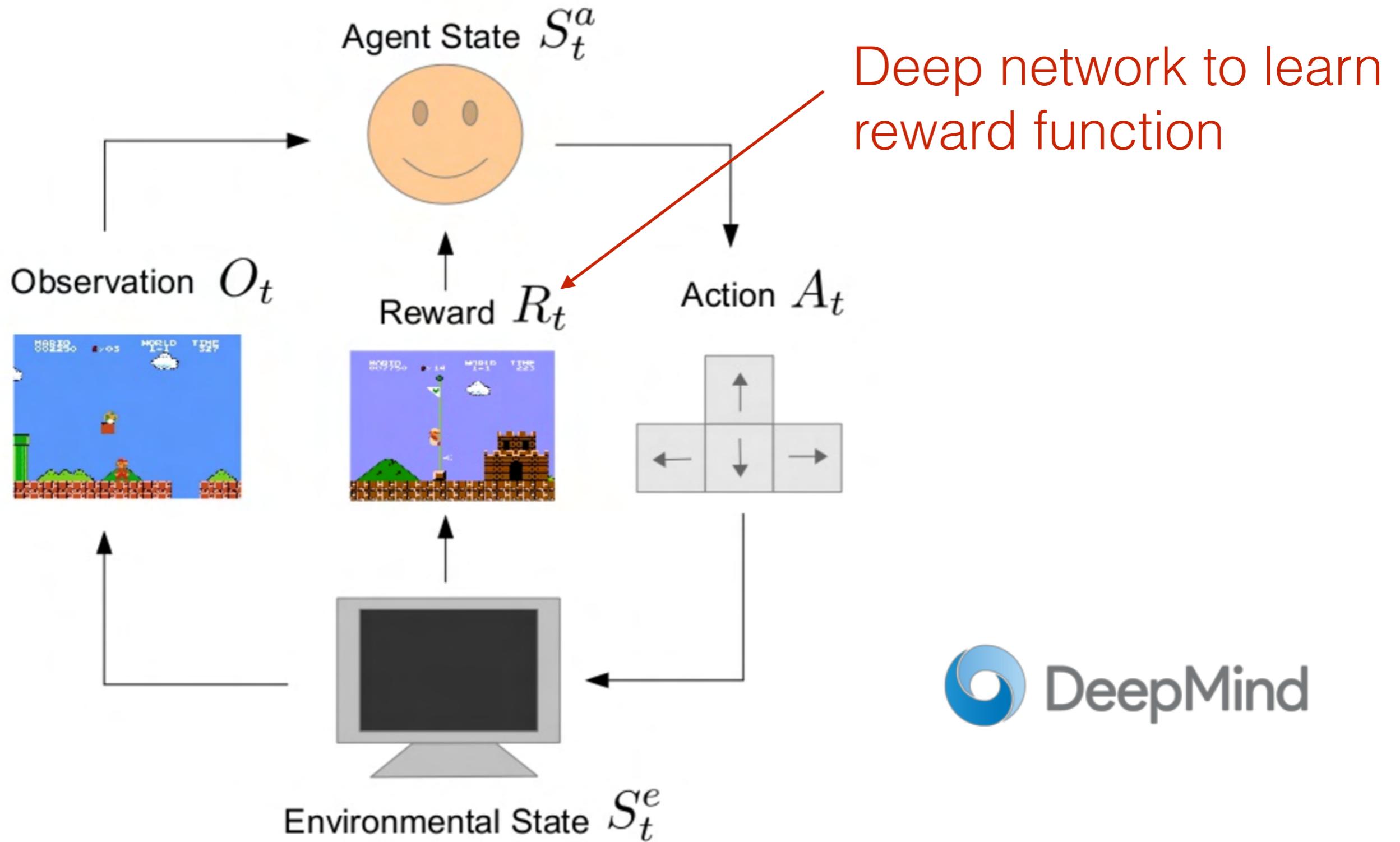
which has a nonzero morphism we may assume that f_i is of finite presentation over S . We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \rightarrow \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $GL_{S'}(x'/S'')$ and we win. \square

To prove study we see that $\mathcal{F}|_U$ is a covering of \mathcal{X}' , and \mathcal{T}_i is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and \mathcal{F}_p exists and let \mathcal{F}_i be a presheaf of \mathcal{O}_X -modules on \mathcal{C} as a \mathcal{F} -module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\text{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1} \mathcal{F})$$

Deep Reinforcement Learning

Reinforcement learning

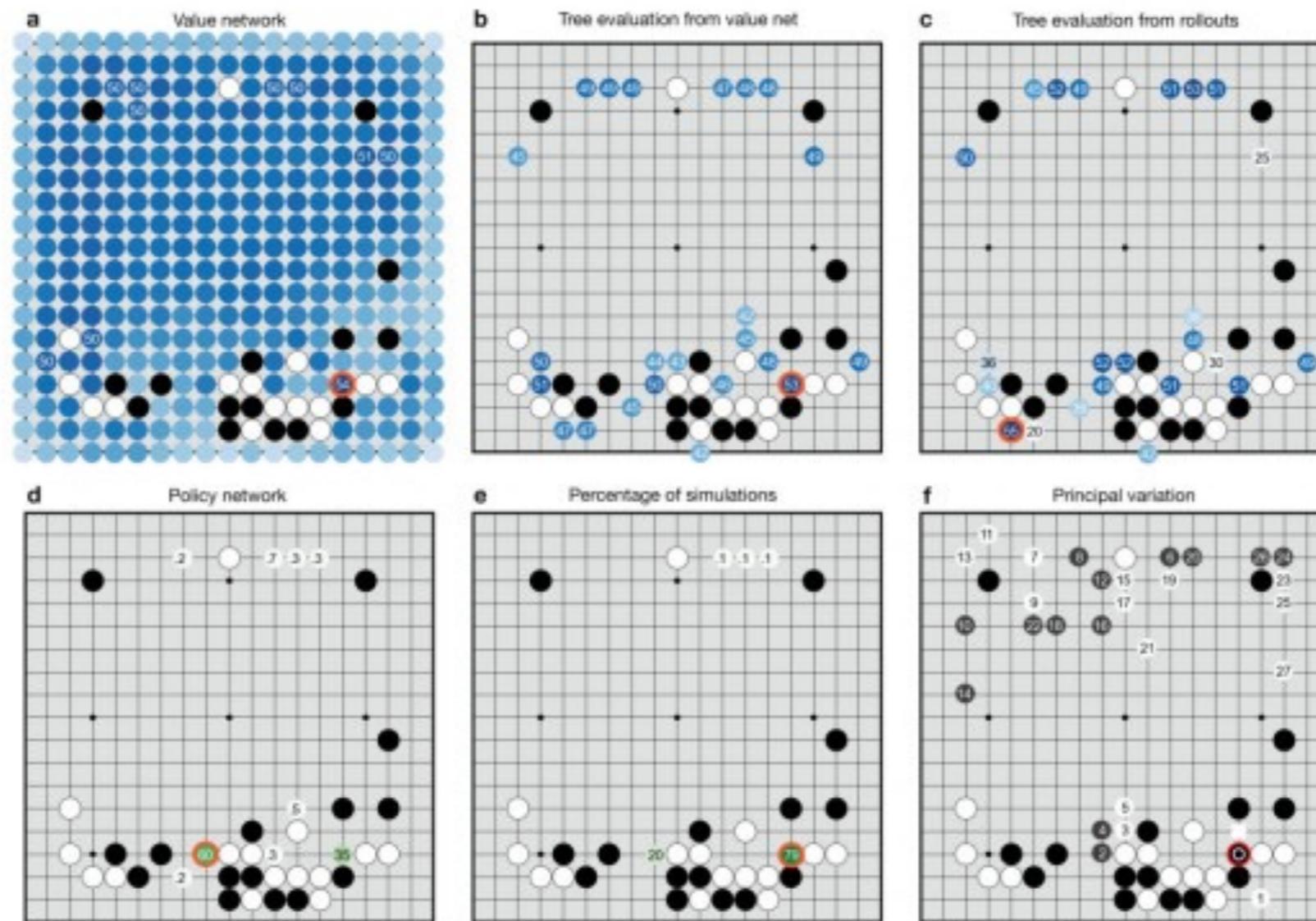


Deep Reinforcement Learning

Reinforcement learning



DeepMind



AlphaGo

Deep Reinforcement Learning

Reinforcement learning

CMU ARTIFICIAL INTELLIGENCE IS TOUGH POKER PLAYER

Libratus builds substantial lead in Brains vs. AI competition

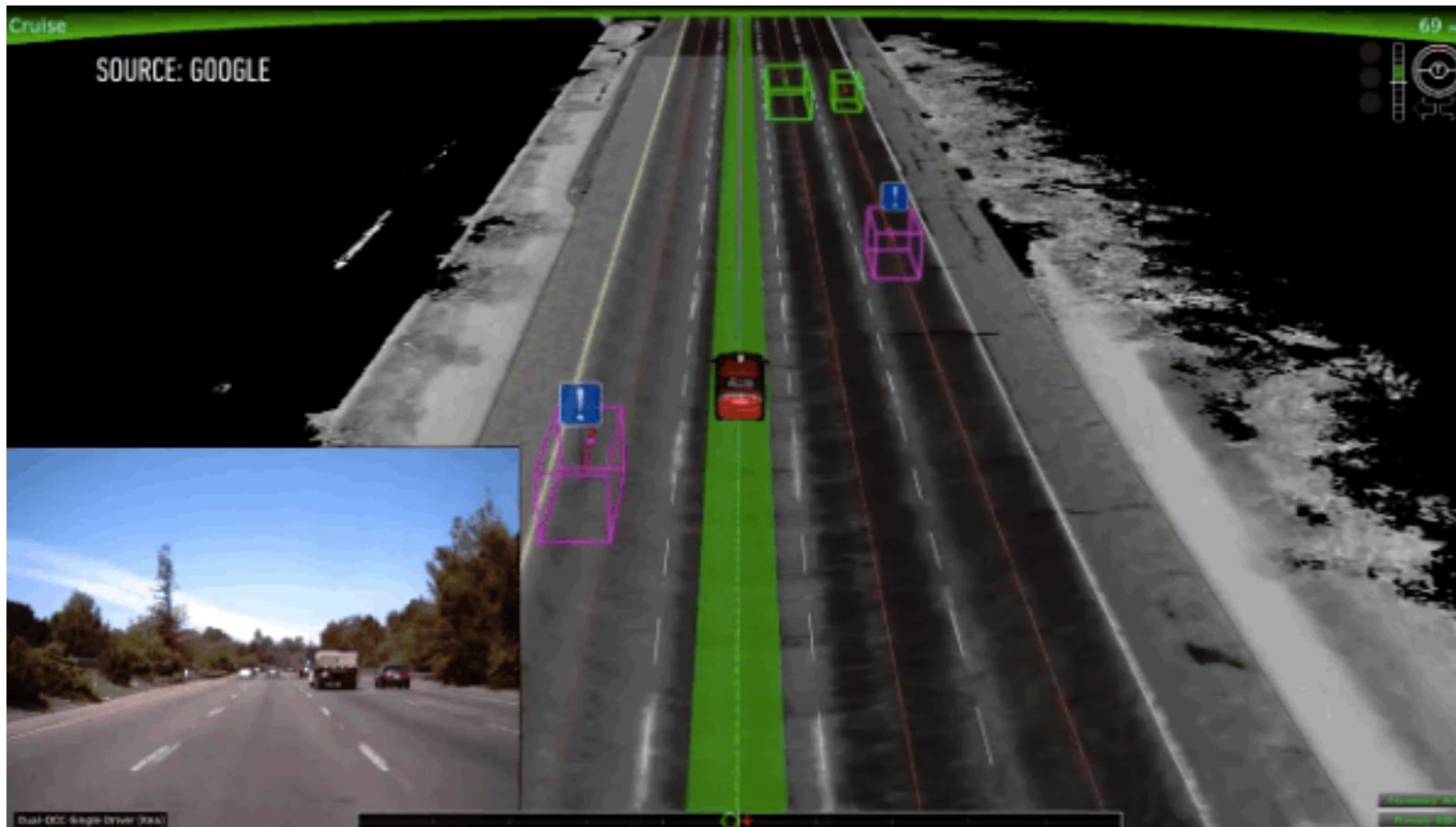
By [Byron Spice](#) (CMU) and [Garrett Allen](#) (Rivers Casino)



Deep Reinforcement Learning

Reinforcement learning + image

Self-driving



Deep Learning Trinity

1. Architecture

Supervised

- Full connected nn
- Convolutional nn
- Recurrent nn
- & thousands variations

Unsupervised

- Autoencoder
- Deep belief network
- Generative Adversarial Net
- & thousands variations

2. Training

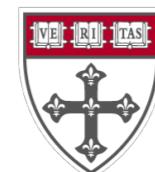
- Stochastic gradient descent
- Dropout
- Pretraining
- & thousands variations

3. Applications

- Many dirty works...

Clustering: K-means

Junwei Lu



HARVARD
T.H. CHAN

SCHOOL OF PUBLIC HEALTH
Department of Biostatistics



Machine Learning

- Supervised learning: Prediction (Y_i, X_i)

Regression, Classification

- Unsupervised learning: Information inside X_i 's

Finding clusters / groups

- We will still look two family of clustering methods

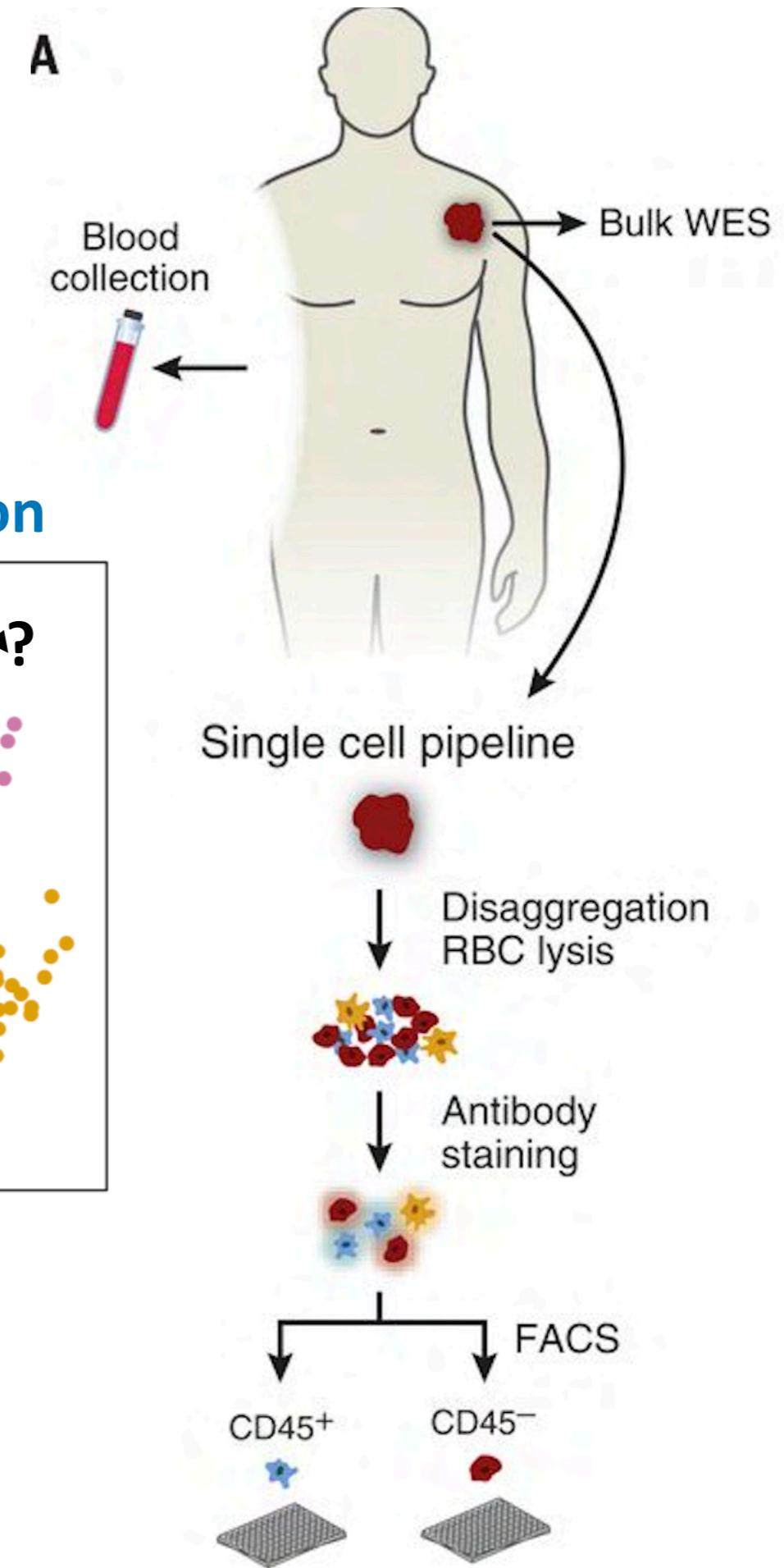
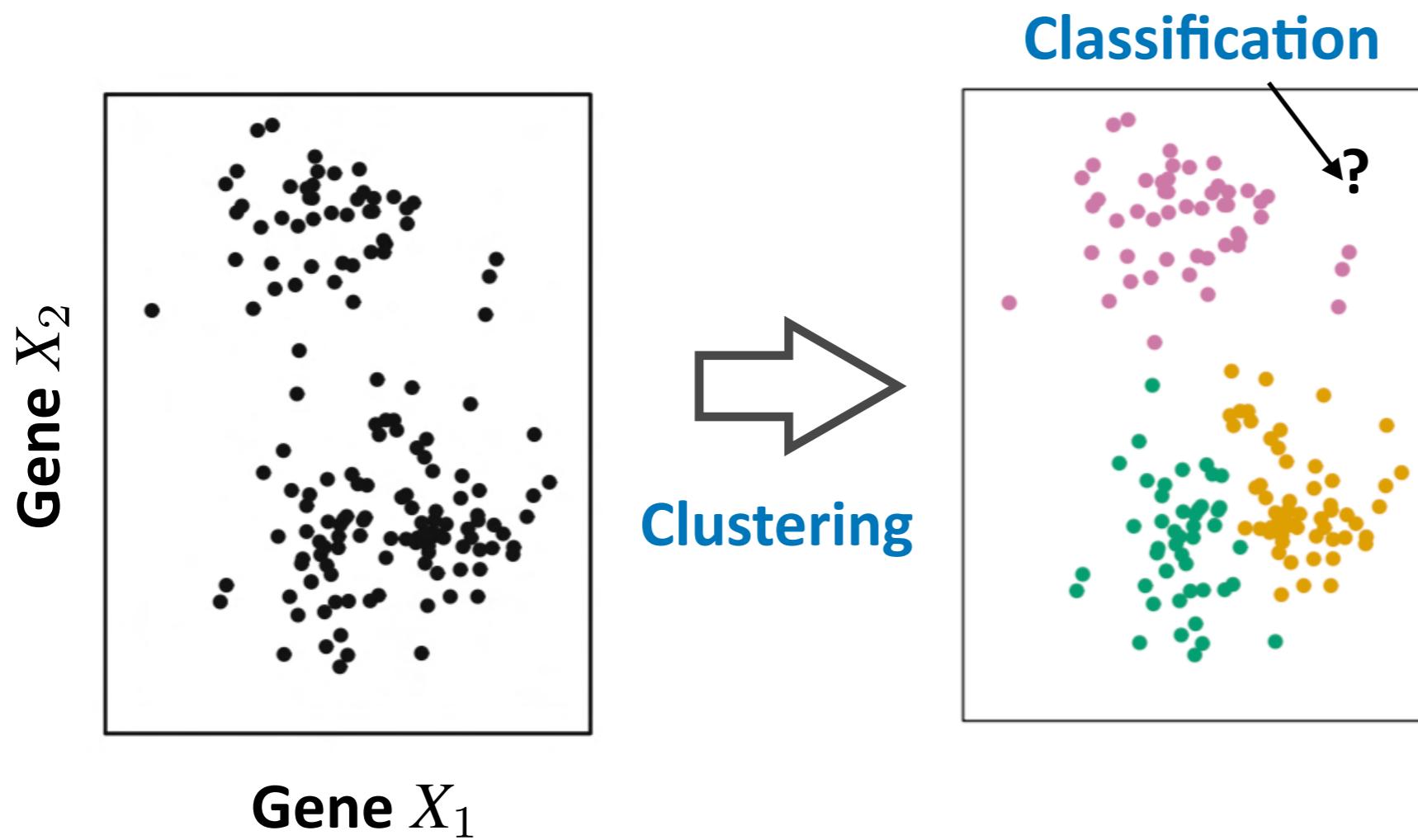
Model-free: K-means clustering

Model-based: EM algorithm

Toy Example

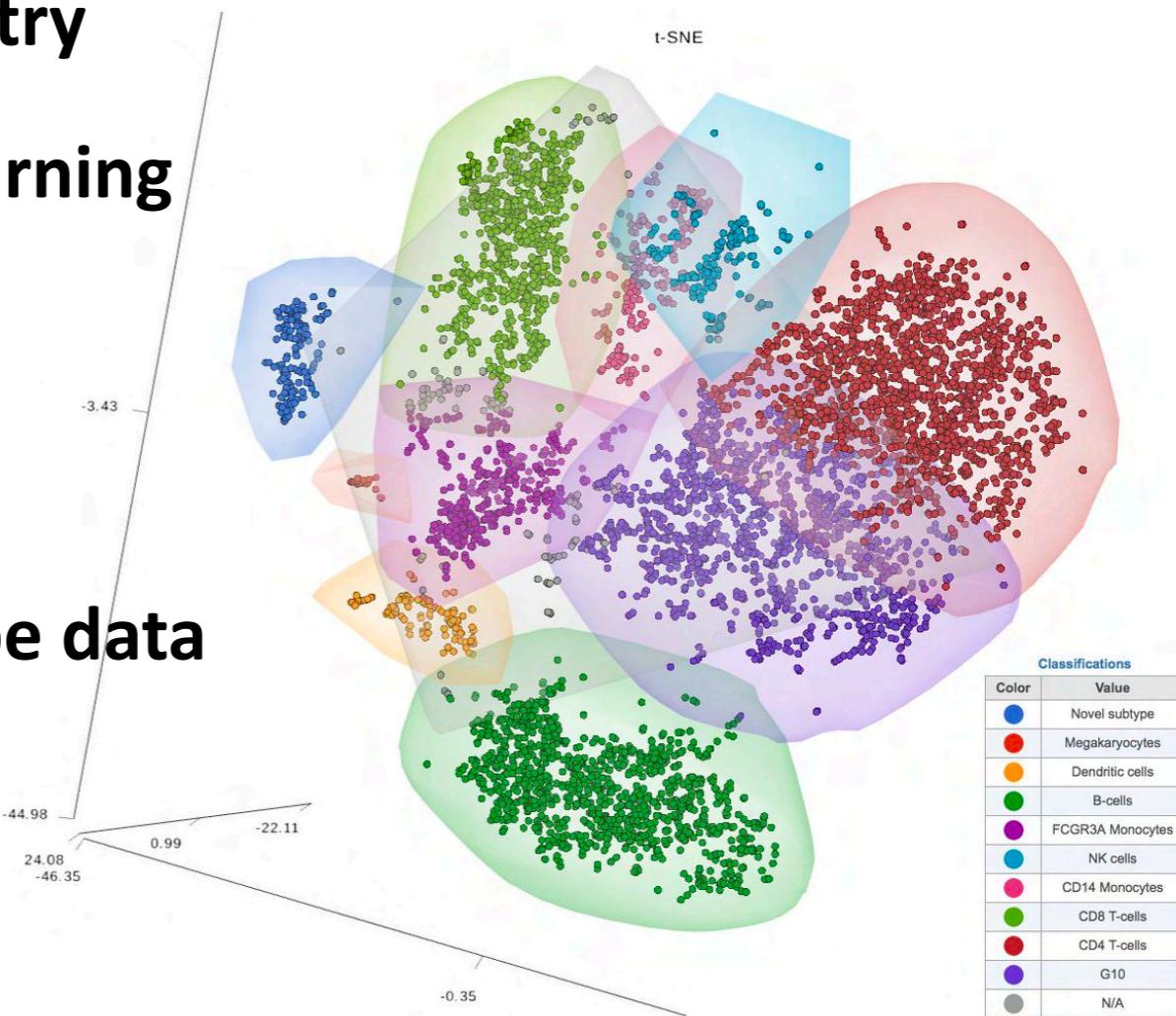
Single-cell RNA expression level data

Goal: Cell types classification vs clustering



Application of Clustering

- Finding hidden structures
 - e.g., discovering new cancer/cell subtypes
- Imputing group labels
 - e.g., gating cell types in flow cytometry
- Feature engineering for supervised learning
 - e.g., distance to cluster centers
- Removing unwanted variation
 - e.g., population structure in genotype data



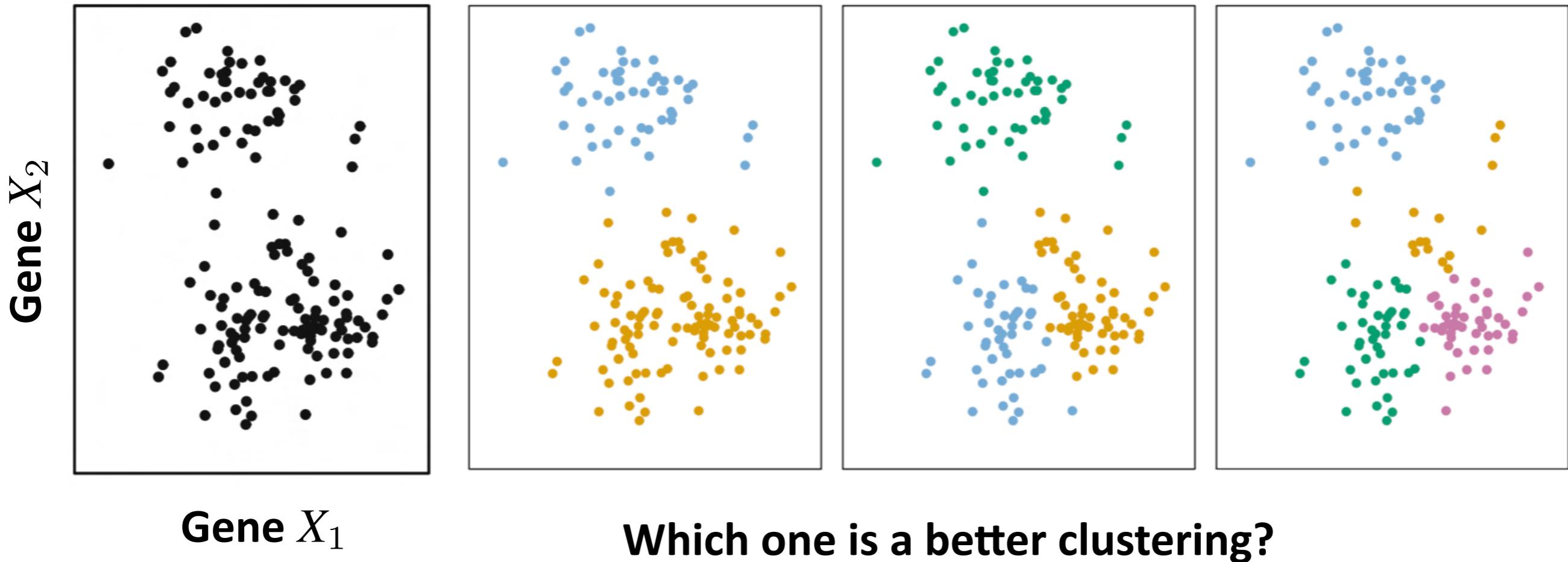
Challenges of Clustering

- Unsupervised learning: No correct answers



Challenges of Clustering

- Unsupervised learning: Number of clusters unknown



Clustering Algorithms

- Model-free methods

- Center based: K-means

- Hierarchical based: hierarchical clustering

- Model-based methods

- Expectation-maximization for mixture models

K-means clustering

- K-means is one of the oldest and most commonly used clustering methods

Intuition: Iteratively assign points to the closest cluster center

K-Means Algorithm (Basic Idea)

Step 1. Initialization by randomly assign the points to K groups

Repeat until the cluster assignment stop changing:

Step2. Set new cluster center μ_k = sample mean of points in group k

Step 3. Reassign each point to the group k with the nearest μ_k

K-means clustering

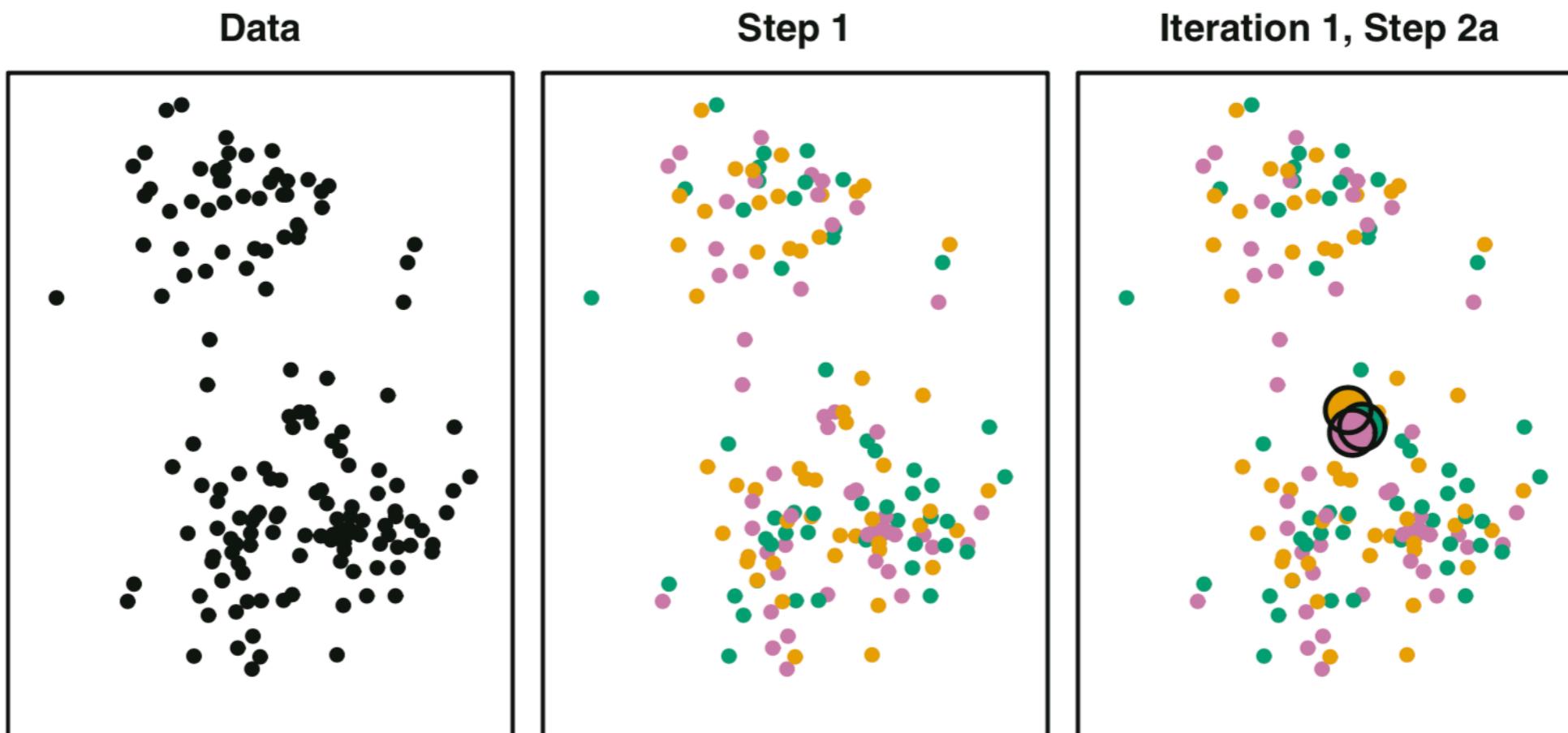
K-Means Algorithm (Basic Idea)

Step 1. Initialization by randomly assign the points to K groups

Repeat until the cluster assignment stop changing:

Step2. Set new cluster center $\mu_k = \text{sample mean of points in group } k$

Step 3. Reassign each point to the group k with the nearest μ_k



K-means clustering

K-Means Algorithm (Basic Idea)

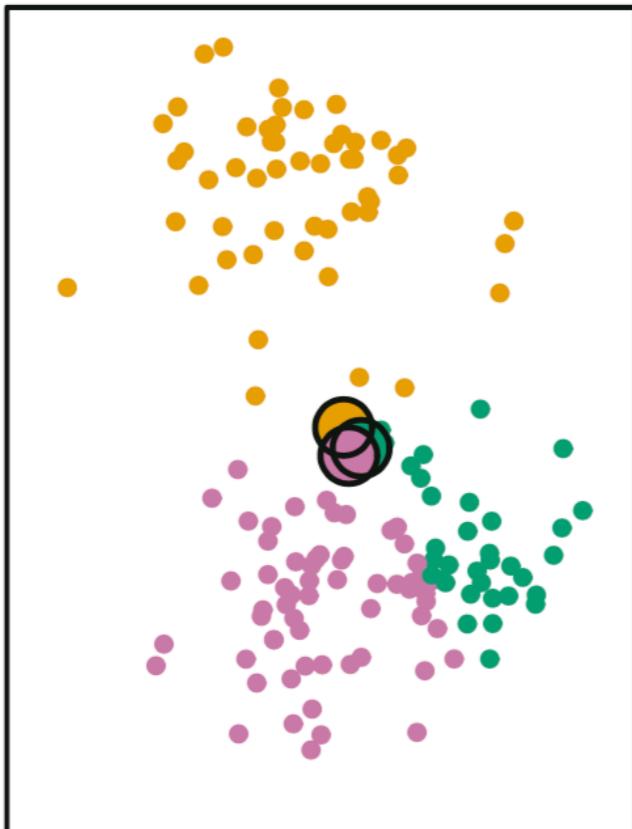
Step 1. Initialization by randomly assign the points to K groups

Repeat until the cluster assignment stop changing:

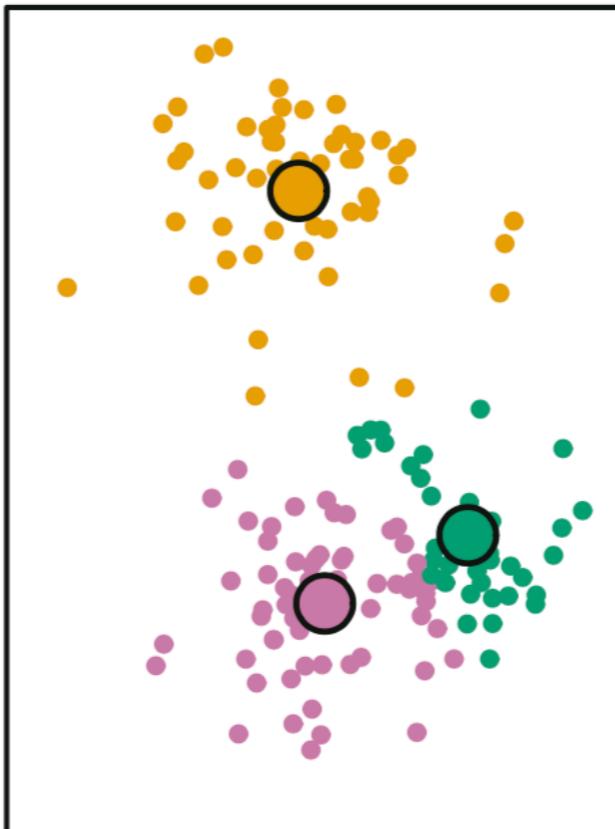
Step2. Set new cluster center $\mu_k = \text{sample mean of points in group } k$

Step 3. Reassign each point to the group k with the nearest μ_k

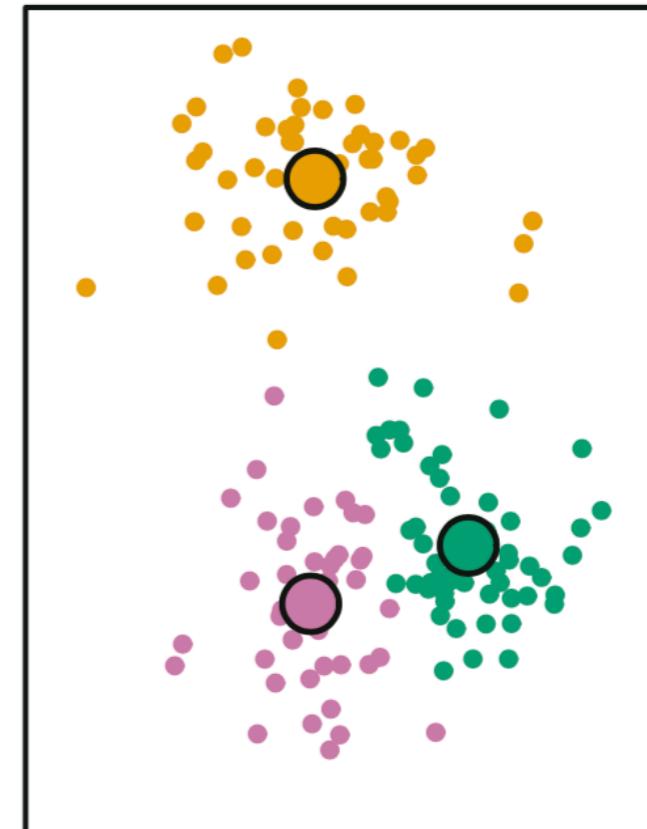
Iteration 1, Step 2b



Iteration 2, Step 2a



Final Results



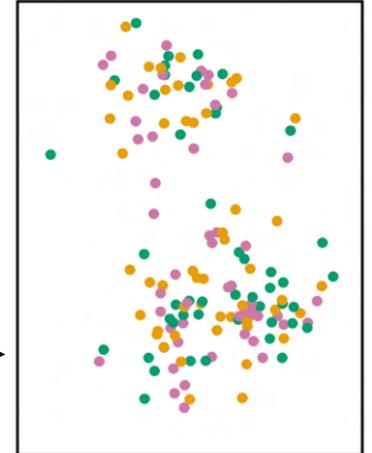
K-means clustering

K-Means Algorithm (Rigorous & Detailed)

Input: Data $x_1, \dots, x_n \in \mathbb{R}^p$, and an integer $K > 0$.

Output: Cluster assignments $c_1, \dots, c_n \in \{1, \dots, K\}$.

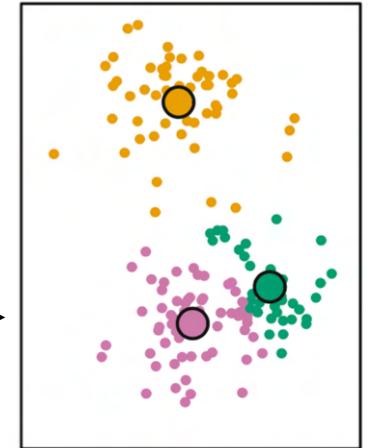
Randomly initialize $c_1, \dots, c_n \in \{1, \dots, K\}$.



Repeat until no change in the c 's is observed:

1. For $k = 1, \dots, K$: **define** $A_k = \{i : c_i = k\}$ **and compute**

$$\mu_k \leftarrow \frac{1}{|A_k|} \sum_{i \in A_k} x_i.$$



2. For $i = 1, \dots, n$: **update** $c_i \leftarrow \operatorname{argmin}_k \|x_i - \mu_k\|$.

K-means clustering



Question: Why choose the center as the sample mean?

Answer: To find the best cluster center closest to all points in the group.

The following two are equivalent:

$$\mu_k \leftarrow \frac{1}{|A_k|} \sum_{i \in A_k} x_i \iff \mu_k \leftarrow \operatorname{argmin}_{\mu} \sum_{i \in A_k} \|x_i - \mu\|$$

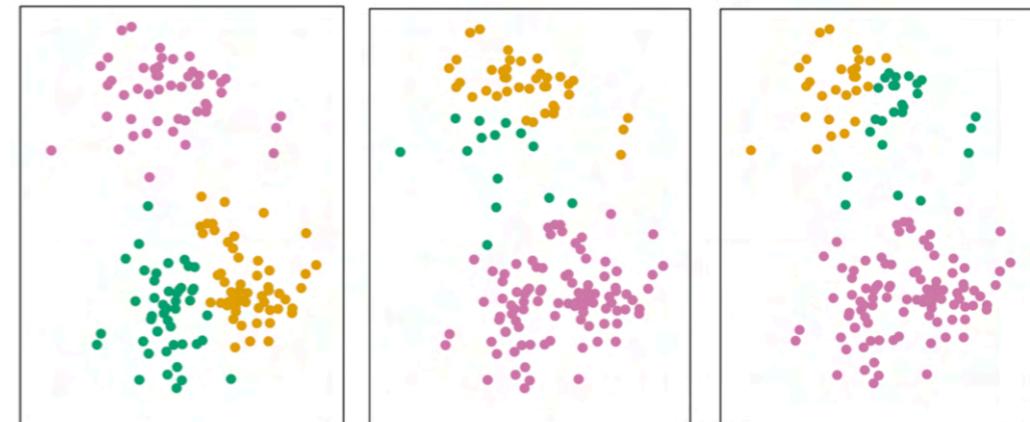
K-means aims to minimize the overall distance to cluster centers

$$F(\mu_{1:K}, c_{1:n}) = \sum_{i=1}^n \|x_i - \mu_{c_i}\|^2.$$



Exponential amount of assignments

Hard to find the global minimum



K-means clustering

K-Means Algorithm (Geometric Intuition)

K-means greedily minimizes the overall distance to cluster centers

$$F(\mu_{1:K}, c_{1:n}) = \sum_{i=1}^n \|x_i - \mu_{c_i}\|^2 = \sum_{k=1}^K \sum_{i:c_i=k} \|x_i - \mu_k\|^2.$$

Original Step. Set new cluster center $\mu_k = \text{sample mean of points in group } k$

Geometric meaning:

Fix assignment $c_{1:n}$, $\mu_k \leftarrow \operatorname{argmin}_\mu \sum_{i:c_i=k} \|x_i - \mu\|^2$ for each k

Original Step: For $i = 1, \dots, n$: update $c_i \leftarrow \operatorname{argmin}_k \|x_i - \mu_k\|$.

Geometric meaning: Fix centers $\mu_{1:k}$, $c_i \leftarrow \operatorname{argmin}_k \|x_i - \mu_k\|$ for each i

K-means clustering

K-Means Algorithm (Geometric Intuition)

K-means greedily minimizes the overall distance to cluster centers

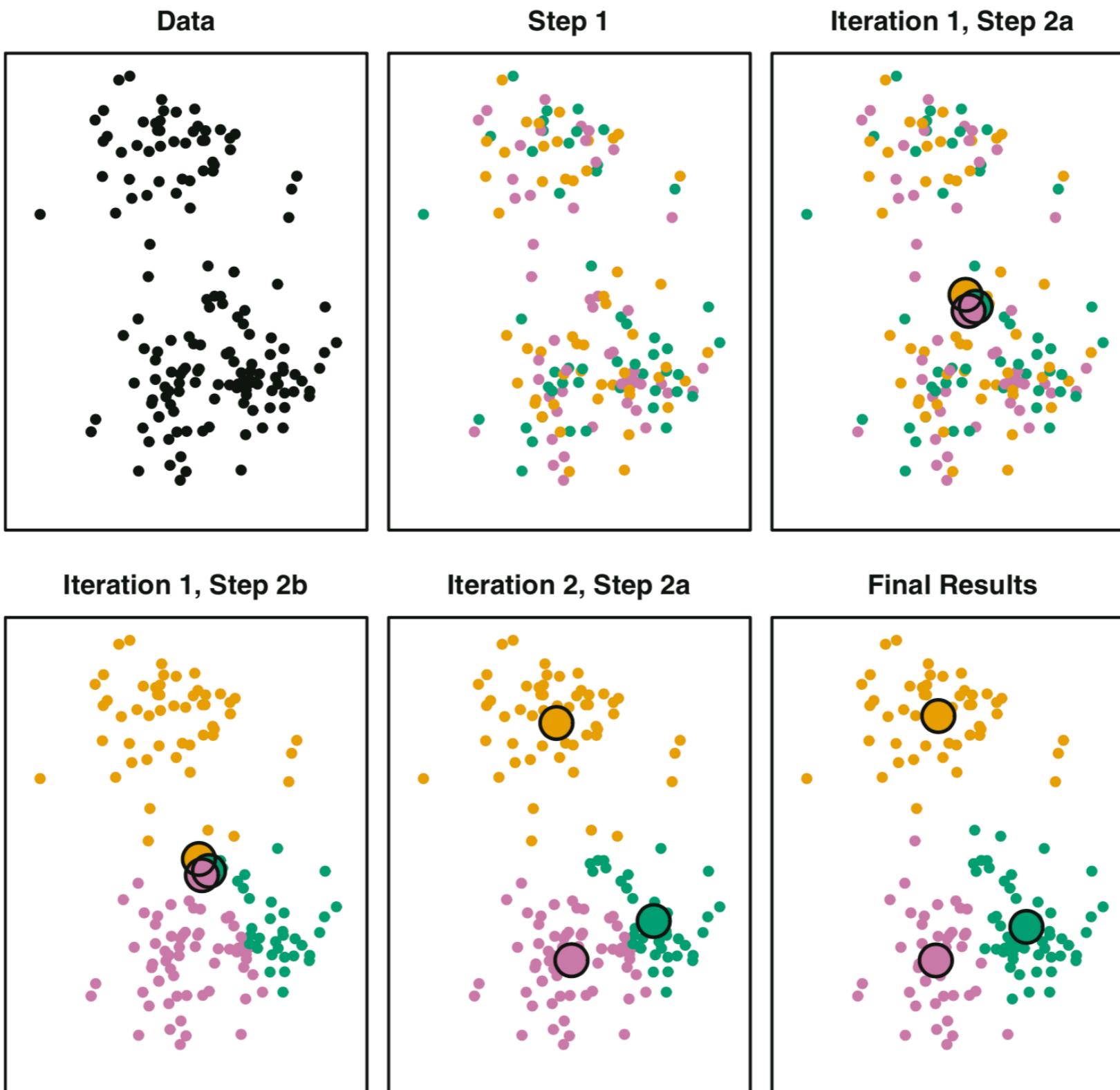
$$F(\mu_{1:K}, c_{1:n}) = \sum_{i=1}^n \|x_i - \mu_{c_i}\|^2 = \sum_{k=1}^K \sum_{i:c_i=k} \|x_i - \mu_k\|^2.$$

K-means essentially alternates between two steps

1. Minimize over $\mu_{1:K}$ with $c_{1:n}$ held fixed.
2. Minimize over $c_{1:n}$ with $\mu_{1:K}$ held fixed.

K-means clustering

K-Means Algorithm (Let us review again)



K-means clustering: Pros and Cons

Pros

- Simple and easy
- Scale to large dimension and sample size
- Typically converges quickly

Cons

- Sometimes converges to a suboptimal solution
- Need to choose the number of clusters K

How to choose the number of clusters

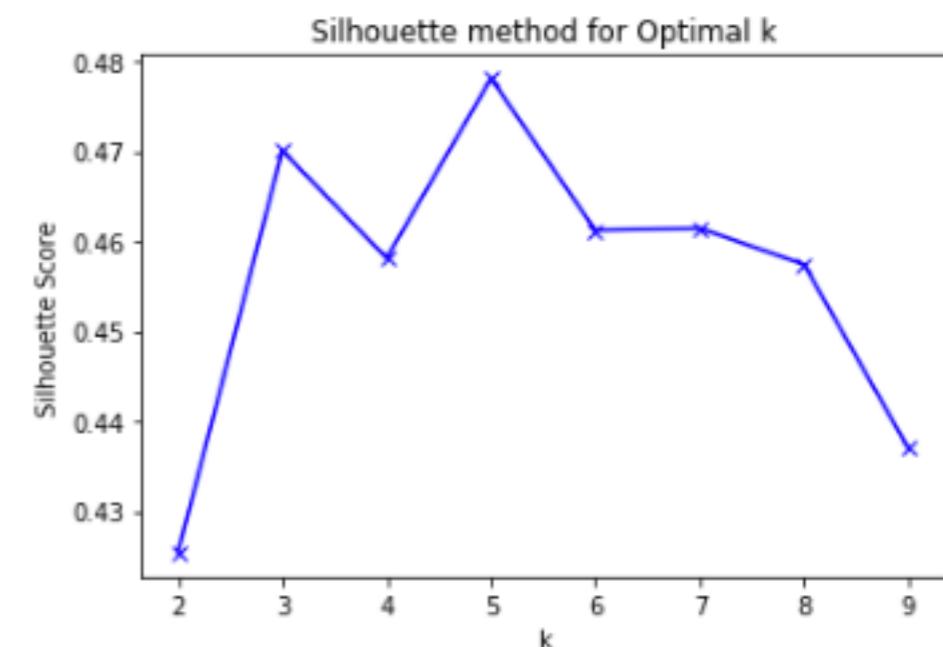
Find the K which maximizes the **silhouette coefficient**

Silhouette coefficient measures the concentration of clusters

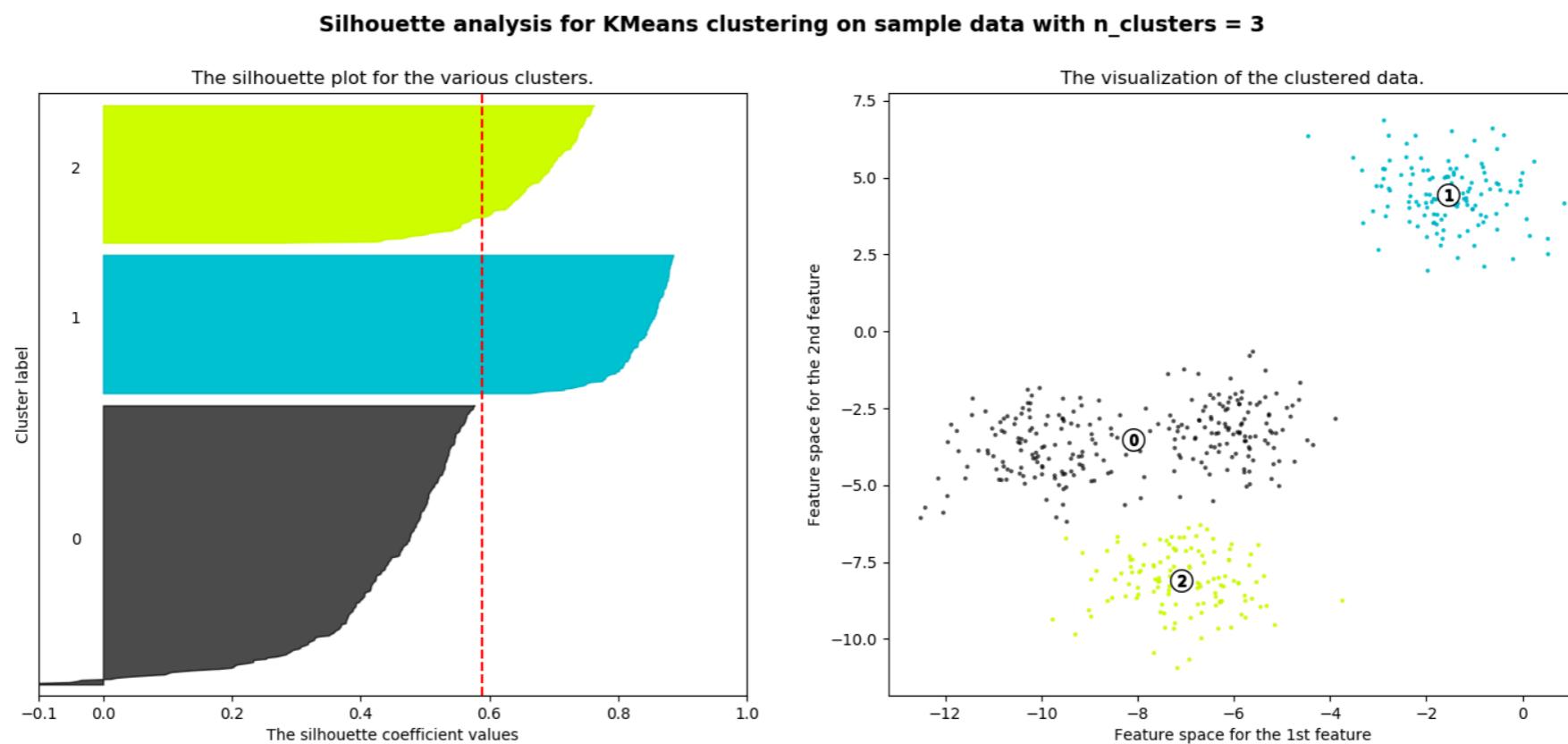
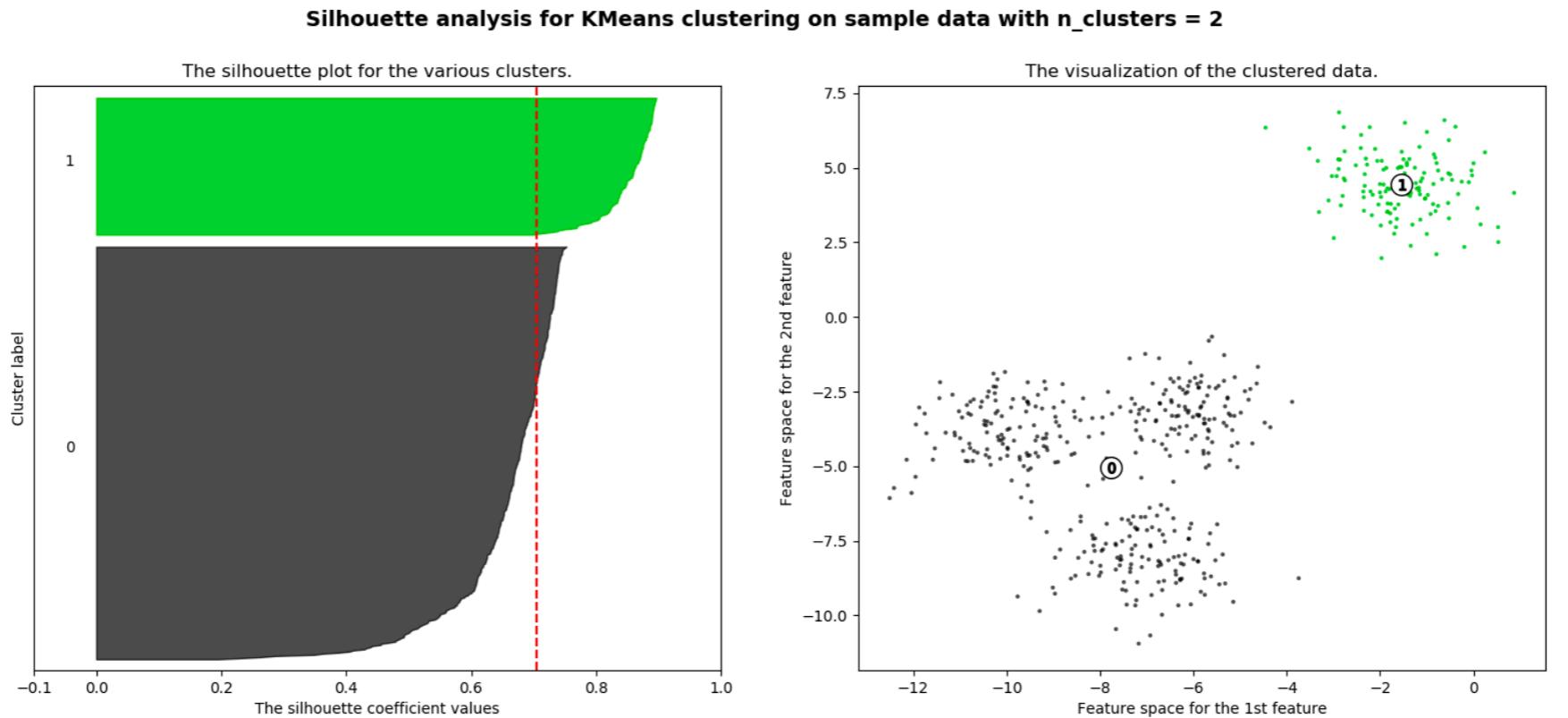
= $(x - y)/\max(x,y)$, where

x = mean nearest cluster distance: mean distance to the instances of the next closest cluster.

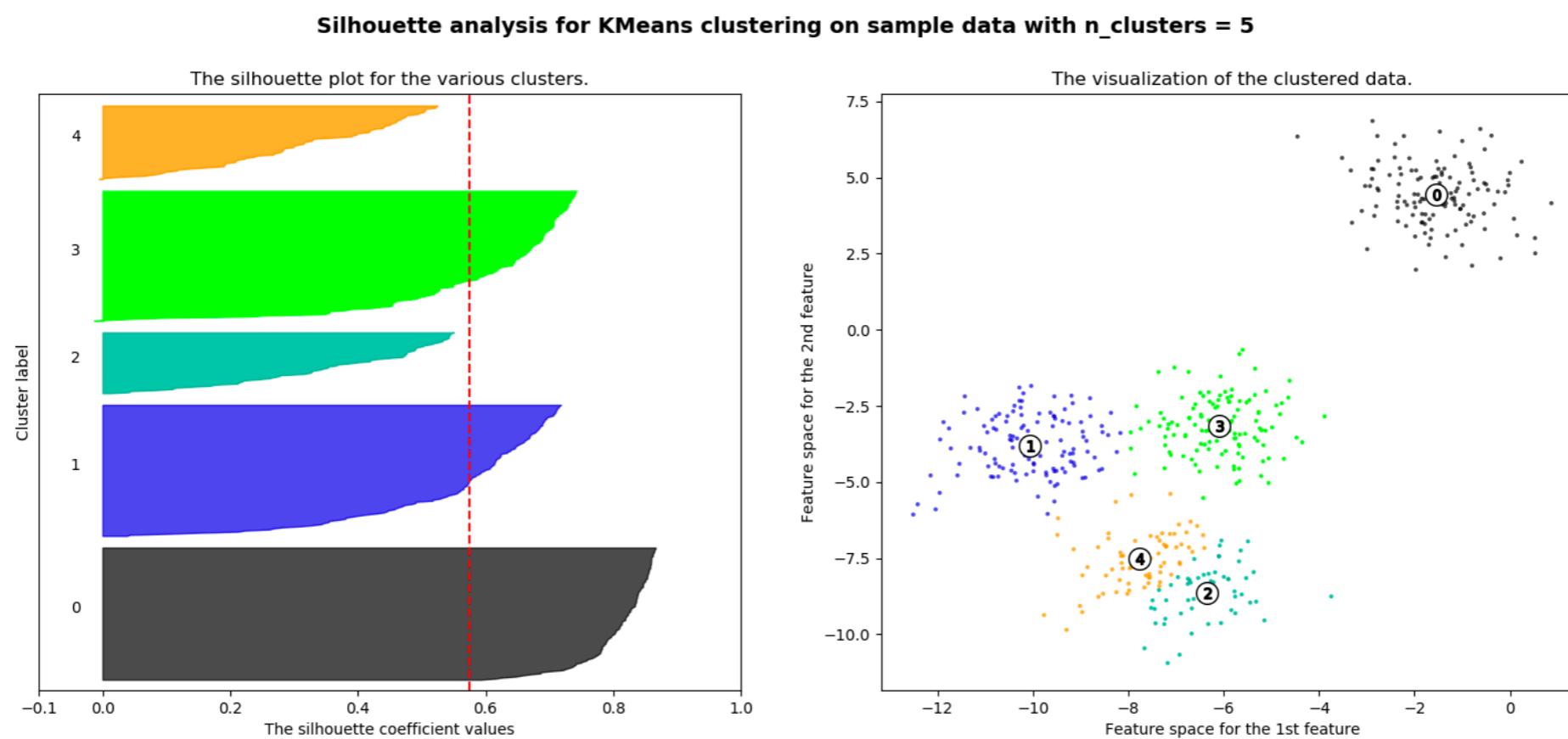
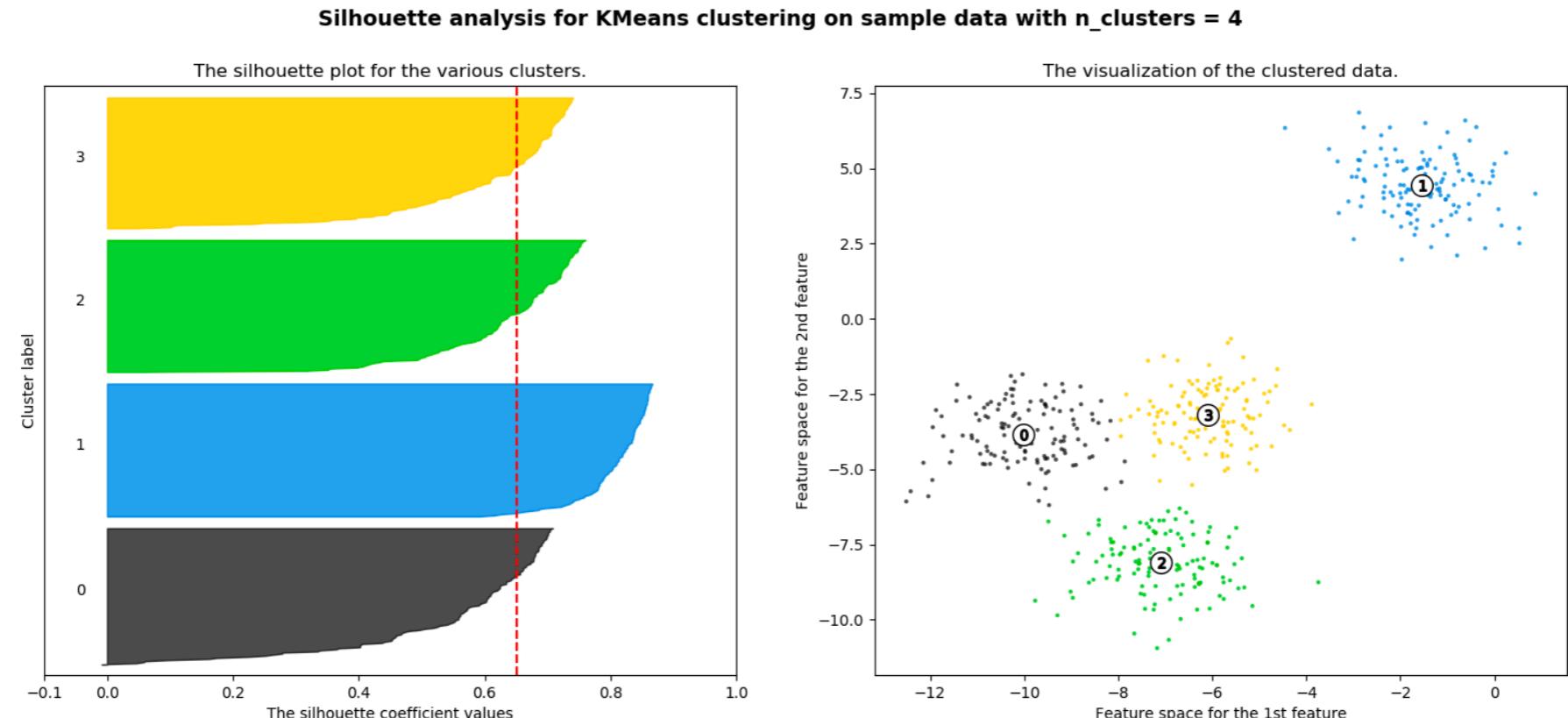
y = mean intra cluster distance: mean distance to the other instances in the same cluster



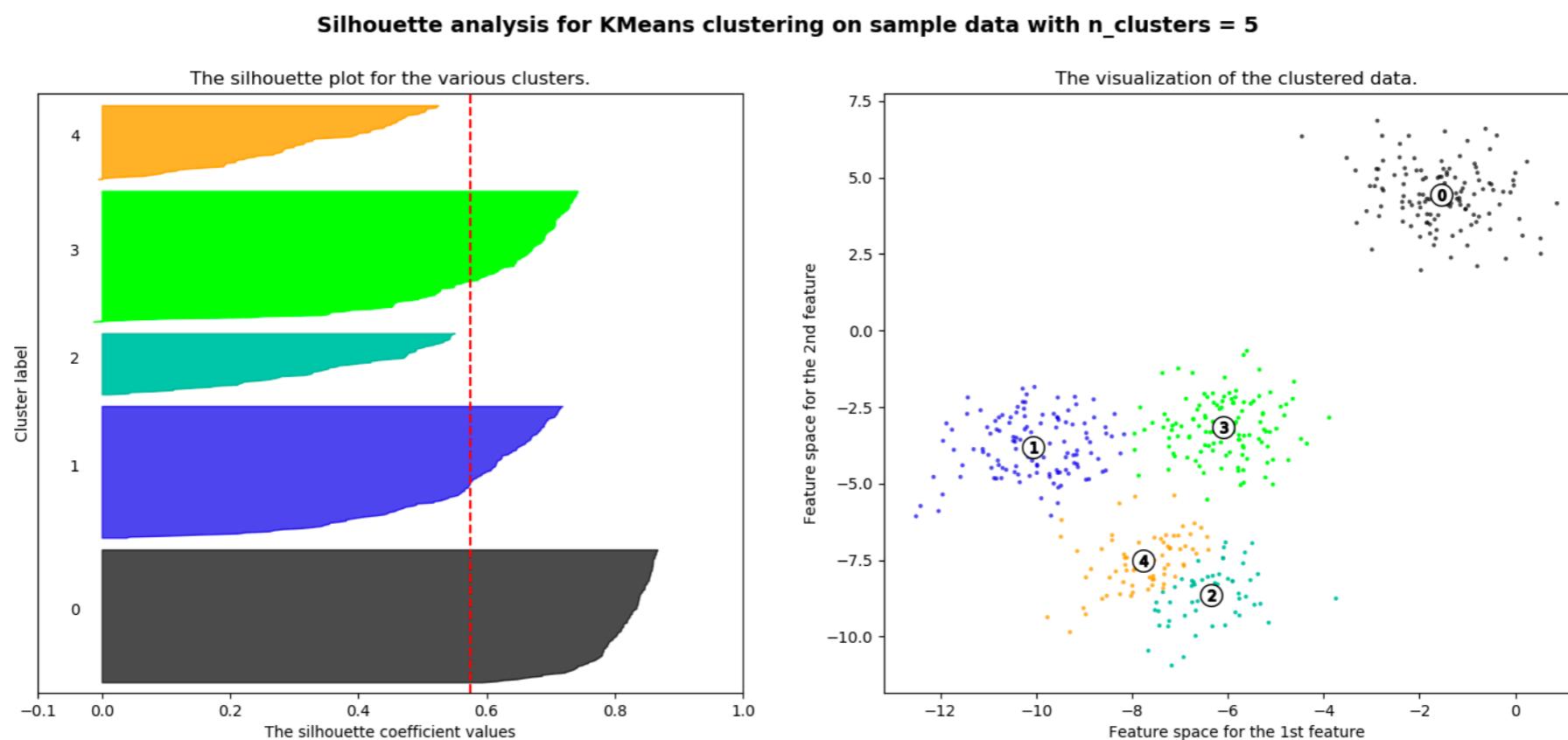
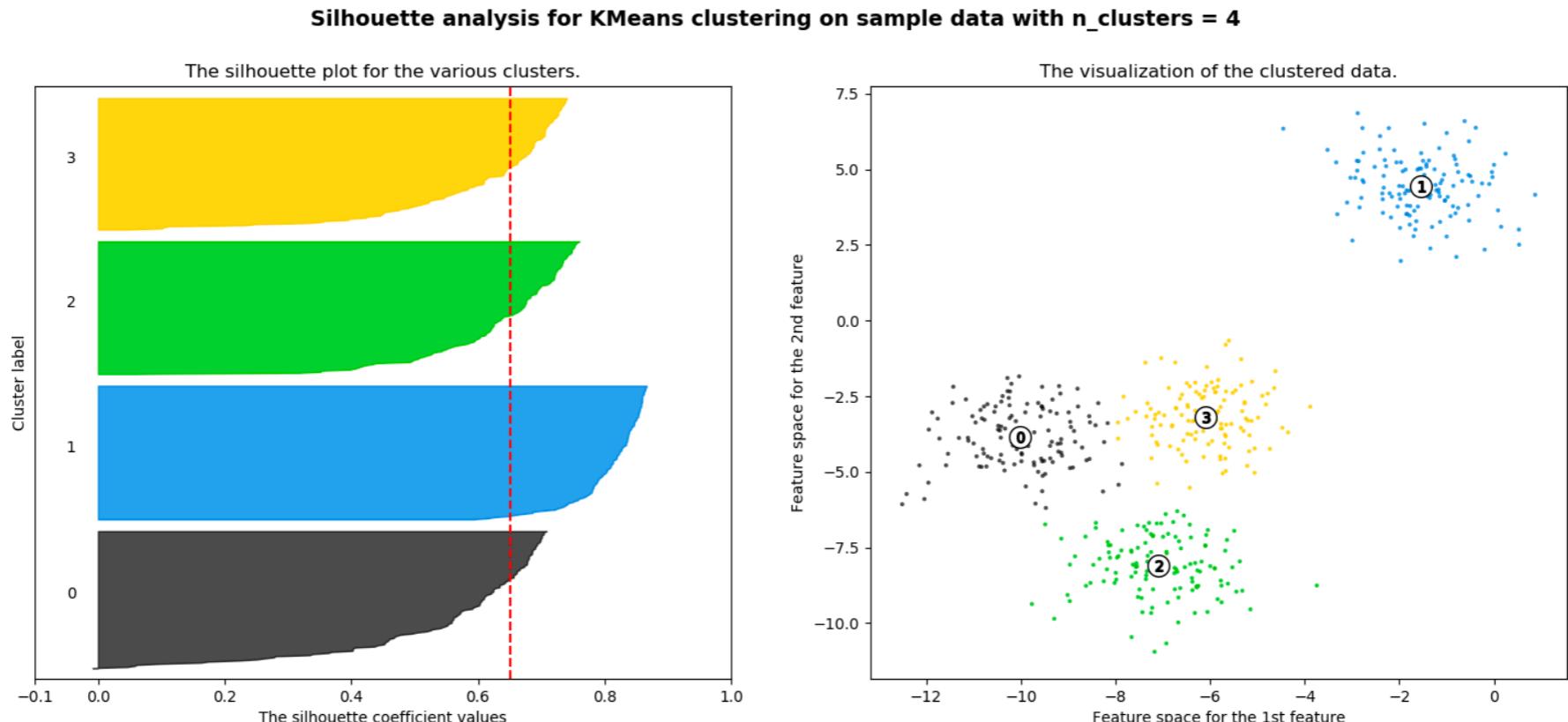
How to choose the number of clusters



How to choose the number of clusters

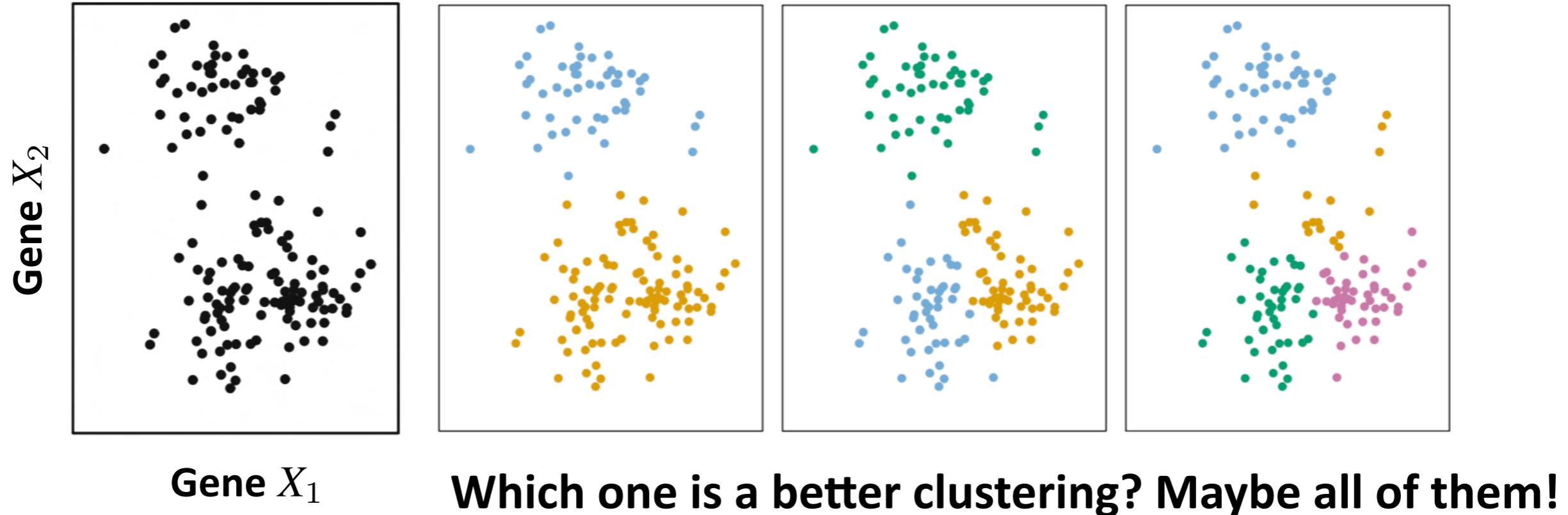


How to choose the number of clusters

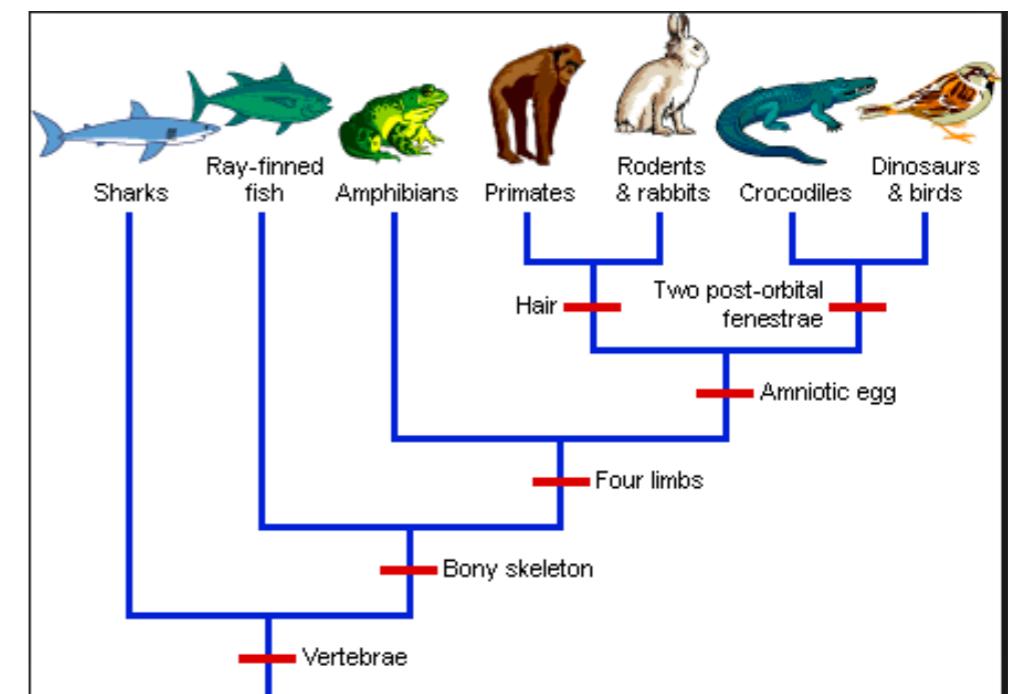


Hierarchical clustering: Introduction

- Hierarchical clustering gives results for all clustering numbers



- It is more popular in genetics
- It yields a dendrogram



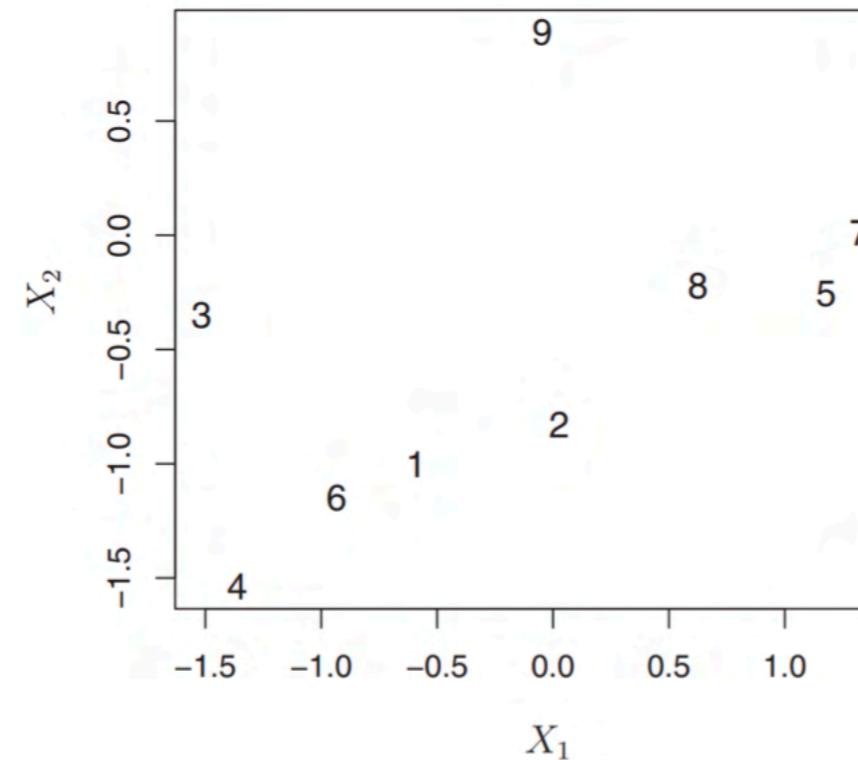
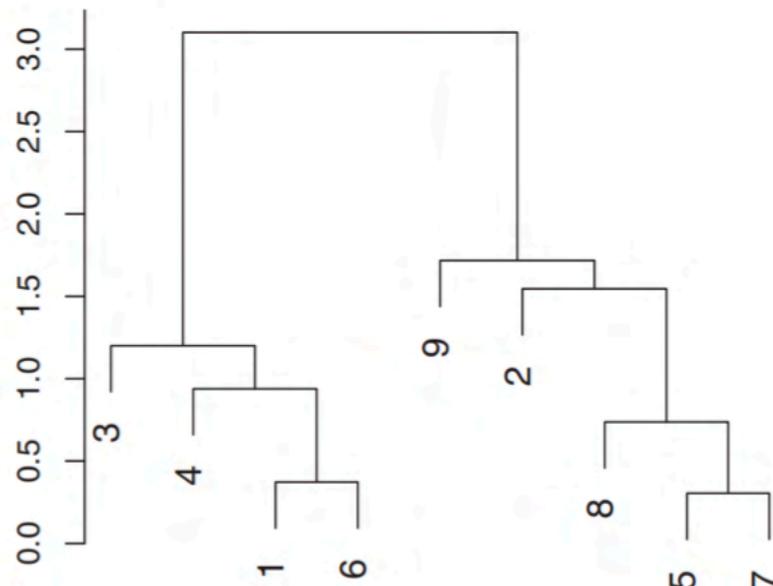
Hierarchical clustering

Hierarchal Clustering (Basic Idea)

Step 1. Every point starts in its own cluster.

Step 2. Find the pair of clusters that are most **similar**, and merge them

Step 3. Repeat step 1 until they all merged together



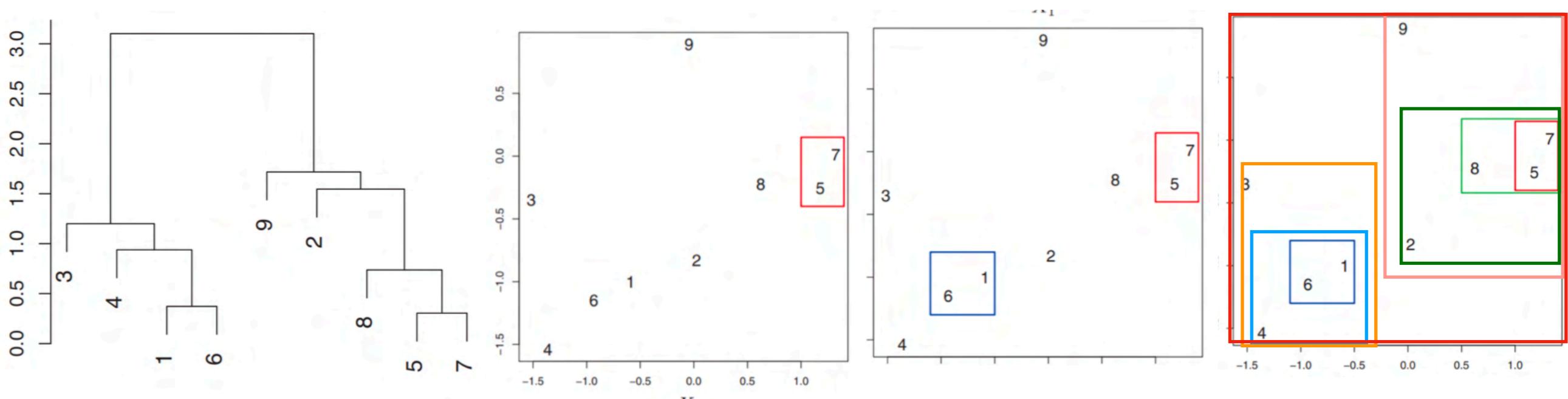
Hierarchical clustering

Hierarchal Clustering (Basic Idea)

Step 1. Every point starts in its own cluster.

Step 2. Find the pair of clusters that are most **similar**, and merge them

Step 3. Repeat step 1 until they all merged together



Hierarchical clustering

Hierarchal Clustering (Detailed)

For $i = n, n - 1, \dots, 2 :$

- 1. Compute the pairwise cluster dissimilarities among the i clusters.**
- 2. Identify the pair of clusters that are least dissimilar**
- 3. Fuse the two clusters in the pair**

- How to measure the dissimilarity between two clusters? Specify
 - The **dissimilarities**/distance between two data points, e.g., euclidean distance
 - The way to **linkage**: How to convert distance between points to the distance between clusters

Hierarchical clustering: Choice of Linkage

<i>Linkage</i>	<i>Description</i>
Complete	Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the <i>largest</i> of these dissimilarities.
Single	Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the <i>smallest</i> of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are fused one-at-a-time.
Average	Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the <i>average</i> of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable <i>inversions</i> .

Hierarchical clustering: Pros and Cons

Pros

- Yields a dendrogram/tree
- Conceptually simple
- Can handle a variety of data types

Cons

- Computationally expensive for large sample size
- We still need to choose K to get the specific cluster

Clustering: EM Algorithm

Junwei Lu



HARVARD
T.H. CHAN

SCHOOL OF PUBLIC HEALTH
Department of Biostatistics



Clustering Algorithms

- Model-free methods

- Center based: K-means

- Hierarchical based: hierarchical clustering

- Model-based methods

- Expectation-maximization for mixture models

Mixture Model

- Recall the mixture model in classification

Flip a coin / Toss a dice: $\mathbb{P}(Y = k) = \eta_k$

Generate observations: $X|Y = k \sim p_k(x)$

- In classification $\underbrace{X_1, \dots, X_n}_{\text{Observed}}, \underbrace{Y_1, \dots, Y_n}_{\text{Observed}} \leftarrow \text{LDA/QDA}$

In clustering $\underbrace{X_1, \dots, X_n}_{\text{Observed}}, \underbrace{Y_1, \dots, Y_n}_{\text{Hidden}}$

Toy Example

1-D Mixture of Two Gaussians:

Flip a coin: $\mathbb{P}(Y = 1) = \eta$ and $\mathbb{P}(Y = 0) = 1 - \eta$

Generate X: $X|Y = 0 \sim N(\mu_0, \sigma_0^2)$

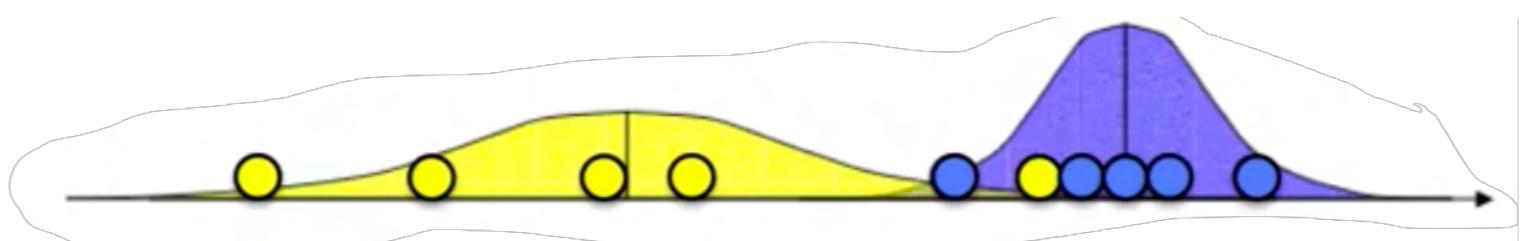
$X|Y = 1 \sim N(\mu_1, \sigma_1^2)$

- In both classification & clustering, estimate $\eta_i = \mathbb{P}(Y_i = 1|X_i)$ and $\mu_0, \mu_1, \sigma_0, \sigma_1, \eta$

What if Y_1, \dots, Y_n are observed?

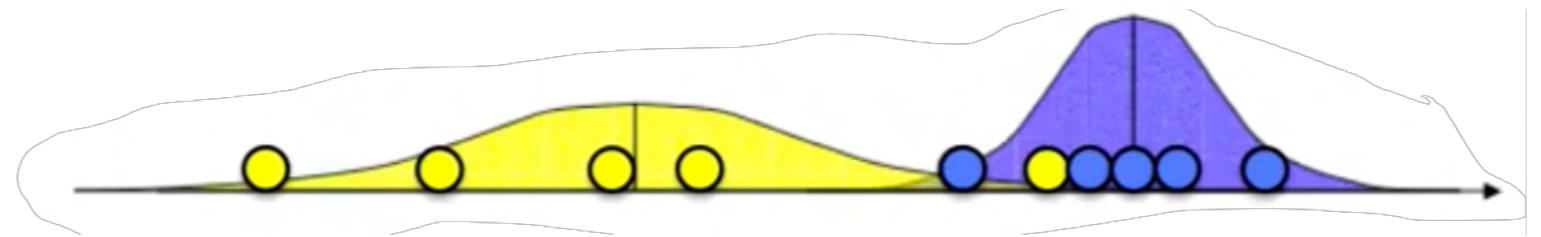
$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:Y_i=k} X_i \quad \hat{\sigma}_k^2 = \frac{1}{n_k} \sum_{i:Y_i=k} (X_i - \hat{\mu}_k)^2$$

$$\hat{\eta} = \frac{1}{n} \sum_{i=1}^n \mathbb{I}(Y_i = i)$$



Toy Example

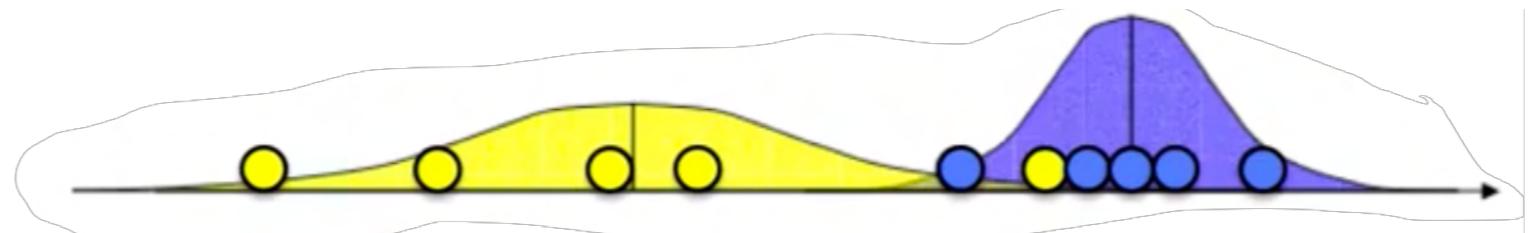
Guess whether point is more likely to be 0 or 1



Toy Example

- In both classification & clustering, estimate $\eta_i = \mathbb{P}(Y_i = 1|X_i)$
What if μ_k, σ_k, η are known? Bayes formula just like LDA/QDA

Guess whether point is more likely to be 0 or 1



Toy Example

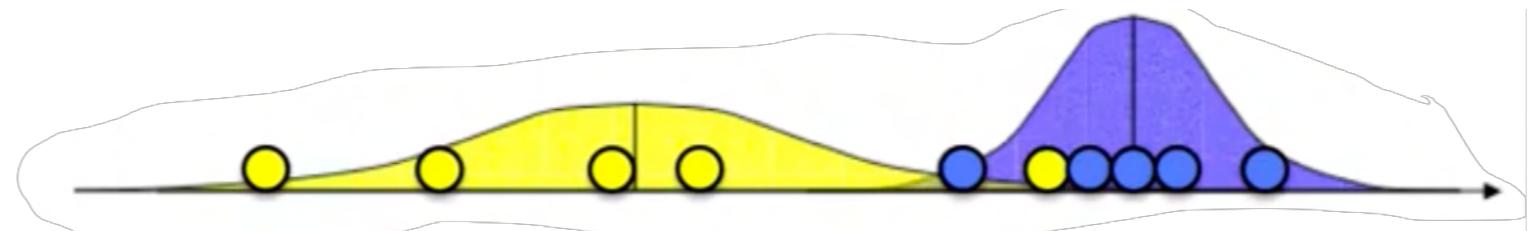
- In both classification & clustering, estimate $\eta_i = \mathbb{P}(Y_i = 1|X_i)$

What if μ_k, σ_k, η are known? Bayes formula just like LDA/QDA

$$\eta_i = \mathbb{P}(Y_i|X_i) = \frac{p(X_i|Y_i = 1)\eta}{p(X_i|Y_i = 1)\eta + p(X_i|Y_i = 0)(1 - \eta)} \quad \text{Bayes}$$

$$p(x|Y = k) = p_k(x) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(x - \mu_k)^2}{2\sigma_k^2}\right)$$

Guess whether point is more likely to be 0 or 1



Alternative Strategy

Recall K-mean is an alternative optimization:

$$F(\mu_{1:K}, c_{1:n}) = \sum_{i=1}^n \|x_i - \mu_{c_i}\|^2 = \sum_{k=1}^K \sum_{i:c_i=k} \|x_i - \mu_k\|^2.$$

1. Minimize over $\mu_{1:K}$ with $c_{1:n}$ held fixed.
2. Minimize over $c_{1:n}$ with $\mu_{1:K}$ held fixed.

Alternative Strategy

Recall K-mean is an alternative optimization:

$$F(\mu_{1:K}, c_{1:n}) = \sum_{i=1}^n \|x_i - \mu_{c_i}\|^2 = \sum_{k=1}^K \sum_{i:c_i=k} \|x_i - \mu_k\|^2.$$

1. Minimize over $\mu_{1:K}$ with $c_{1:n}$ held fixed.
2. Minimize over $c_{1:n}$ with $\mu_{1:K}$ held fixed.

For Gaussian mixture:

Fix Y_i 's, we can get μ_k, σ_k, η

Fix μ_k, σ_k, η 's, we can get $\mathbb{P}(Y_i|X_i)$

Expectation-Maximization (EM) Algorithm: Idea is alternation again!

EM Algorithm

Expectation-Maximization (EM) Algorithm:

Initialization. Randomly initialize μ_k, σ_k, η

E-step: For each point, compute $\mathbb{P}(Y_i|X_i)$

Guess whether point is more likely to be 0 or 1

M-step: Based on new assignment, adjust parameters μ_k, σ_k, η

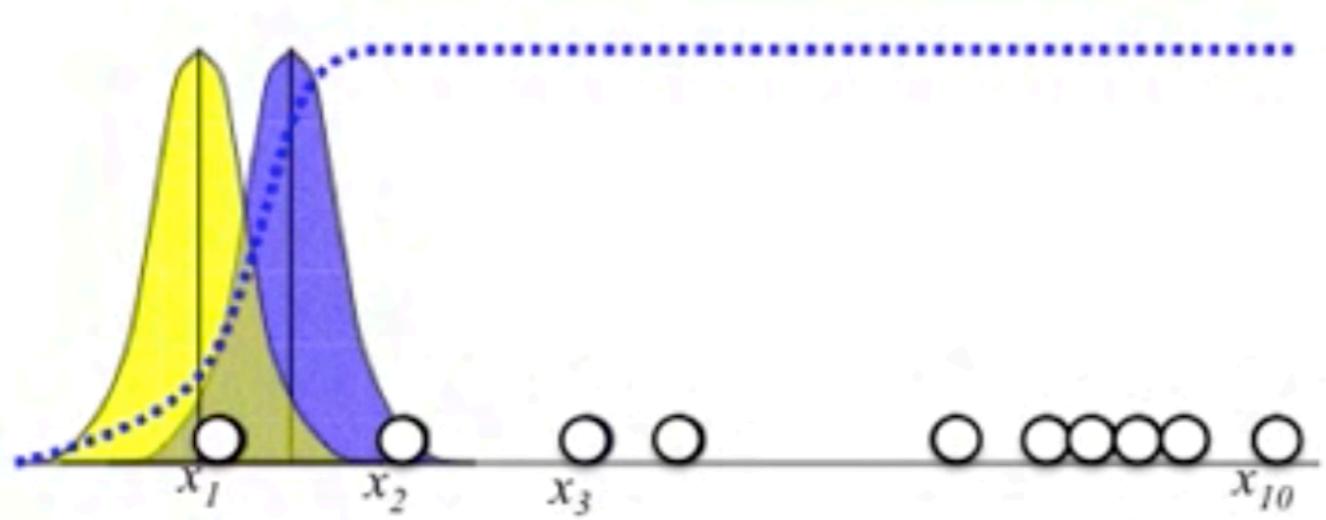
Iterate E&M steps until converges

EM Algorithm (Example)

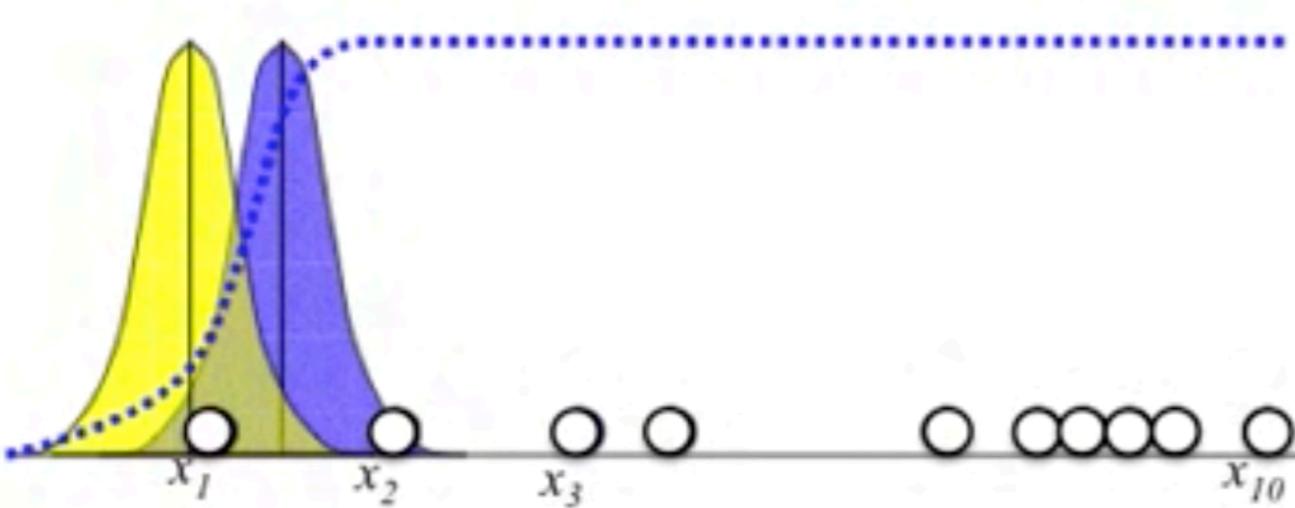
x_1 x_2 x_3 \circ

\circ $\circ\circ\circ$ x_{10}

EM Algorithm (Example)



EM Algorithm (Example)

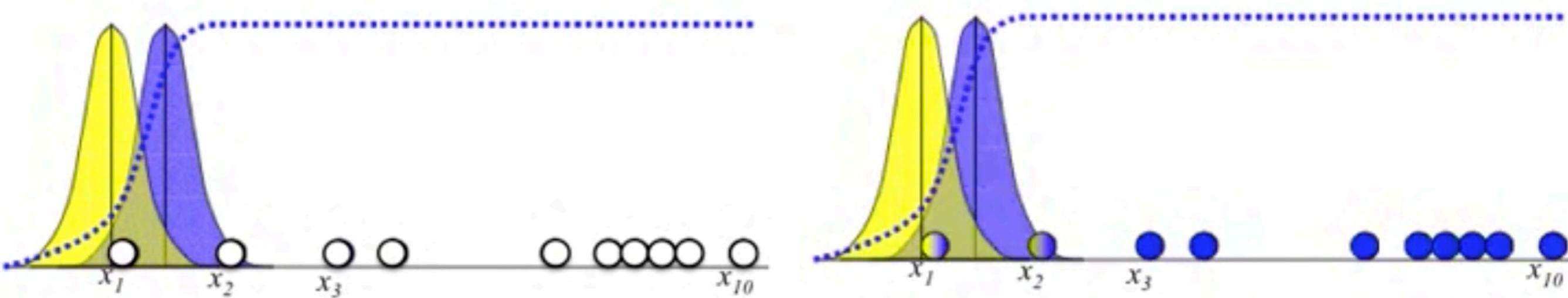


E-step: For each point, compute

$$\eta_i = \mathbb{P}(Y_i|X_i) = \frac{p(X_i|Y_i = 1)\eta}{p(X_i|Y_i = 1)\eta + p(X_i|Y_i = 0)(1 - \eta)} \quad \text{Bayes}$$

$$p(x|Y = k) = p_k(x) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(x - \mu_k)^2}{2\sigma_k^2}\right)$$

EM Algorithm (Example)

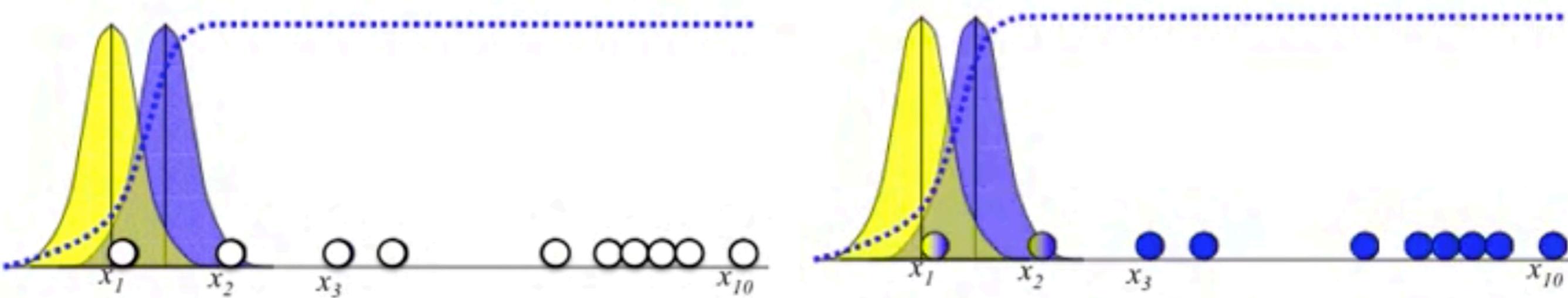


E-step: For each point, compute

$$\eta_i = \mathbb{P}(Y_i|X_i) = \frac{p(X_i|Y_i = 1)\eta}{p(X_i|Y_i = 1)\eta + p(X_i|Y_i = 0)(1 - \eta)} \quad \text{Bayes}$$

$$p(x|Y = k) = p_k(x) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(x - \mu_k)^2}{2\sigma_k^2}\right)$$

EM Algorithm (Example)



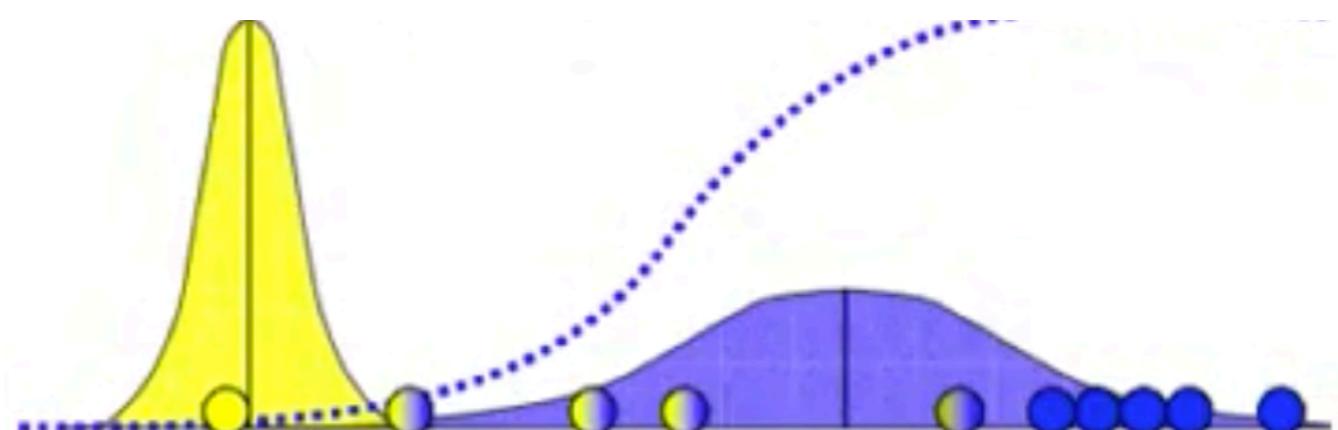
E-step: For each point, compute

$$\eta_i = \mathbb{P}(Y_i|X_i) = \frac{p(X_i|Y_i = 1)\eta}{p(X_i|Y_i = 1)\eta + p(X_i|Y_i = 0)(1 - \eta)} \quad \text{Bayes}$$

$$p(x|Y = k) = p_k(x) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(x - \mu_k)^2}{2\sigma_k^2}\right)$$

M-step: Based on new assignment, adjust parameters

$$\hat{\mu}_k = \frac{1}{\sum_{i=1}^n \hat{\eta}_i} \sum_{i:Y_i=k} \hat{\eta}_i X_i$$



Clustering: EM Algorithm

Junwei Lu



HARVARD
T.H. CHAN

SCHOOL OF PUBLIC HEALTH
Department of Biostatistics



Clustering Algorithms

- Model-free methods

- Center based: K-means

- Hierarchical based: hierarchical clustering

- Model-based methods

- Expectation-maximization for mixture models

Mixture Model

- Recall the mixture model in classification

Flip a coin / Toss a dice: $\mathbb{P}(Y = k) = \eta_k$

Generate observations: $X|Y = k \sim p_k(x)$

- In classification $\underbrace{X_1, \dots, X_n}_{\text{Observed}}, \underbrace{Y_1, \dots, Y_n}_{\text{Observed}} \leftarrow \text{LDA/QDA}$

In clustering $\underbrace{X_1, \dots, X_n}_{\text{Observed}}, \underbrace{Y_1, \dots, Y_n}_{\text{Hidden}}$

Toy Example

1-D Mixture of Two Gaussians:

Flip a coin: $\mathbb{P}(Y = 1) = \eta$ and $\mathbb{P}(Y = 0) = 1 - \eta$

Generate X: $X|Y = 0 \sim N(\mu_0, \sigma_0^2)$

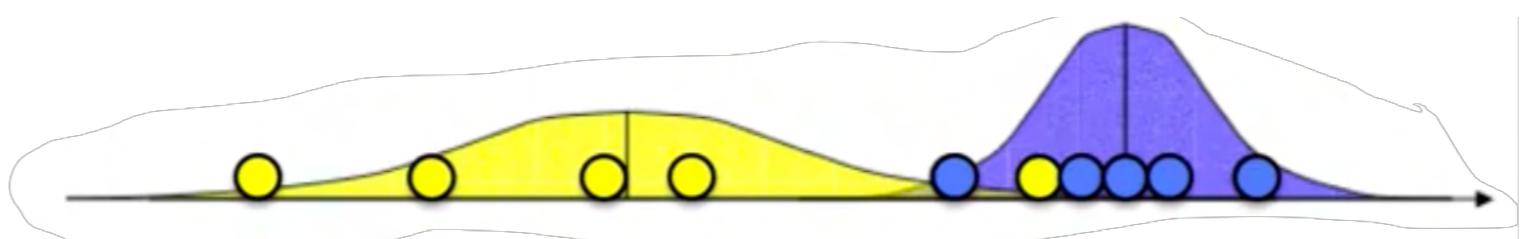
$X|Y = 1 \sim N(\mu_1, \sigma_1^2)$

- In both classification & clustering, estimate $\eta_i = \mathbb{P}(Y_i = 1|X_i)$ and $\mu_0, \mu_1, \sigma_0, \sigma_1, \eta$

What if Y_1, \dots, Y_n are observed? MLE just like in LDA/QDA

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:Y_i=k} X_i \quad \hat{\sigma}_k^2 = \frac{1}{n_k} \sum_{i:Y_i=k} (X_i - \hat{\mu}_k)^2$$

$$\hat{\eta} = \frac{1}{n} \sum_{i=1}^n \mathbb{I}(Y_i = 1)$$



Toy Example

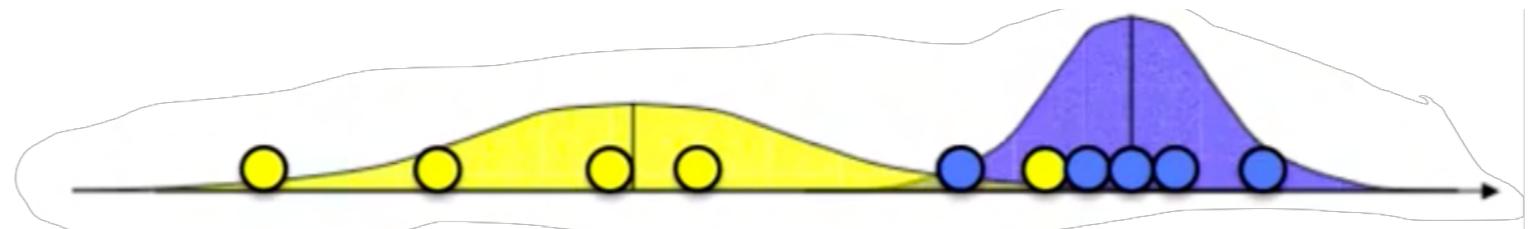
- In both classification & clustering, estimate $\eta_i = \mathbb{P}(Y_i = 1|X_i)$

What if μ_k, σ_k, η are known? Bayes formula just like LDA/QDA

$$\eta_i = \mathbb{P}(Y_i|X_i) = \frac{p(X_i|Y_i = 1)\eta}{p(X_i|Y_i = 1)\eta + p(X_i|Y_i = 0)(1 - \eta)} \quad \text{Bayes}$$

$$p(x|Y = k) = p_k(x) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(x - \mu_k)^2}{2\sigma_k^2}\right)$$

Guess whether point is more likely to be 0 or 1



Alternative Strategy

Recall K-mean is an alternative optimization:

$$F(\mu_{1:K}, c_{1:n}) = \sum_{i=1}^n \|x_i - \mu_{c_i}\|^2 = \sum_{k=1}^K \sum_{i:c_i=k} \|x_i - \mu_k\|^2.$$

1. Minimize over $\mu_{1:K}$ with $c_{1:n}$ held fixed.
2. Minimize over $c_{1:n}$ with $\mu_{1:K}$ held fixed.

For Gaussian mixture:

Fix Y_i 's, we can get μ_k, σ_k, η

Fix μ_k, σ_k, η 's, we can get $\mathbb{P}(Y_i|X_i)$

Expectation-Maximization (EM) Algorithm: Idea is alternation again!

EM Algorithm

Expectation-Maximization (EM) Algorithm:

Initialization. Randomly initialize μ_k, σ_k, η

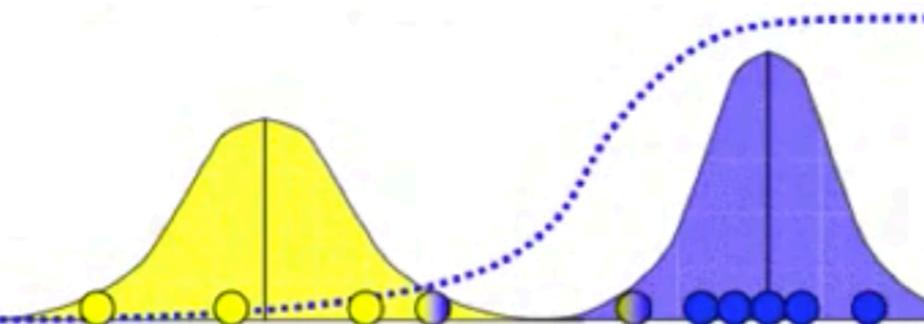
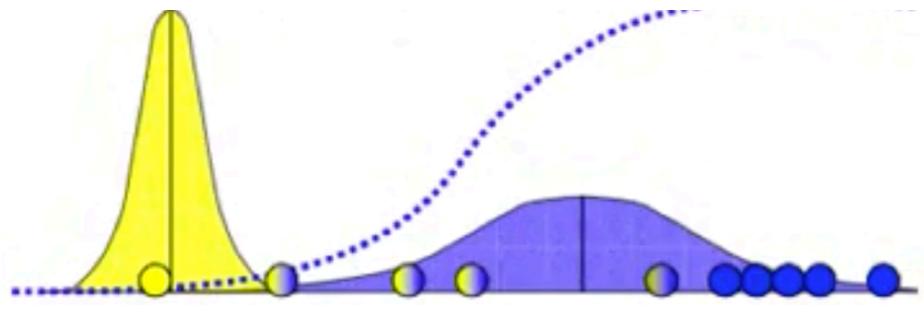
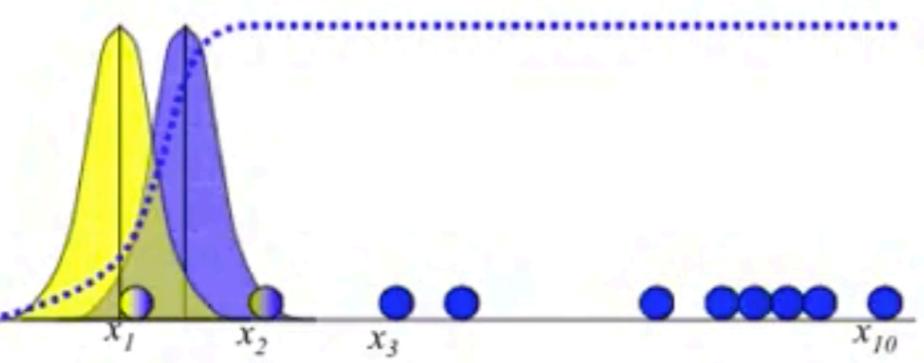
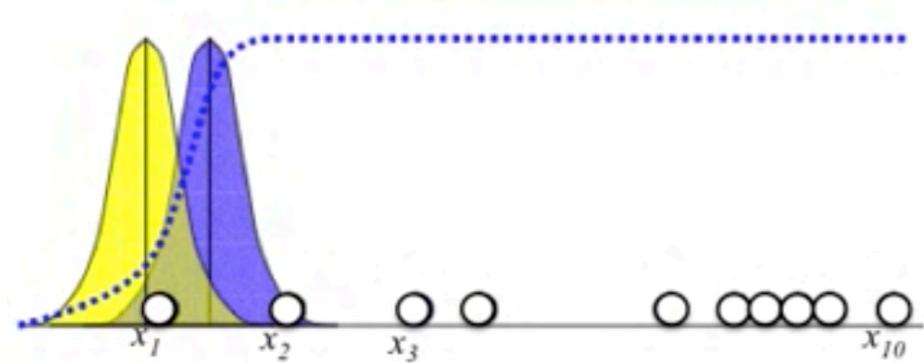
E-step: For each point, compute $\mathbb{P}(Y_i|X_i)$

Guess whether point is more likely to be 0 or 1

M-step: Based on new assignment, adjust parameters μ_k, σ_k, η

Iterate E&M steps until converges

EM Algorithm (1-D Example)



Initialization. Randomly initialize

$$\hat{\mu}_k, \hat{\sigma}_k, \hat{\eta}$$

E-step: Update “guess” of points

$$\hat{\eta}_i = \frac{p(X_i|Y_i = 1)\hat{\eta}}{p(X_i|Y_i = 1)\hat{\eta} + p(X_i|Y_i = 0)(1 - \hat{\eta})}$$

$$p(x|Y = k) = \frac{1}{\sqrt{2\pi\hat{\sigma}_k^2}} \exp\left(-\frac{(x - \hat{\mu}_k)^2}{2\hat{\sigma}_k^2}\right)$$

M-step: Update parameters

$$\hat{\mu}_1 = \frac{\sum_{i=1}^n \hat{\eta}_i X_i}{\sum_{i=1}^n \hat{\eta}_i} \quad \hat{\sigma}_1^2 = \frac{\sum_{i=1}^n \hat{\eta}_i (X_i - \hat{\mu}_1)^2}{\sum_{i=1}^n \hat{\eta}_i}$$

$$\hat{\mu}_0 = \frac{\sum_{i=1}^n (1 - \hat{\eta}_i) X_i}{\sum_{i=1}^n (1 - \hat{\eta}_i)}$$

$$\hat{\sigma}_0^2 = \frac{\sum_{i=1}^n (1 - \hat{\eta}_i) (X_i - \hat{\mu}_0)^2}{\sum_{i=1}^n (1 - \hat{\eta}_i)}$$

$$\hat{\eta} = \frac{1}{n} \sum_{i=1}^n \hat{\eta}_i$$

EM Algorithm

Expectation-Maximization (K mixture Gaussian):

Initialization. Randomly initialize $\hat{\mu}_k^{(0)}, \hat{\sigma}_k^{(0)}, \hat{\eta}^{(0)}$ for $k = 1, \dots, K$

For $t = 0, 1, 2, \dots$

E-step: **for** $k = 1, \dots, K$, $\hat{\eta}_{ik}^{(t+1)} \leftarrow \frac{p_k(X_i)\hat{\eta}_k^{(t)}}{\sum_{k=1}^K p_k(X_i)\hat{\eta}_k^{(t)}}$

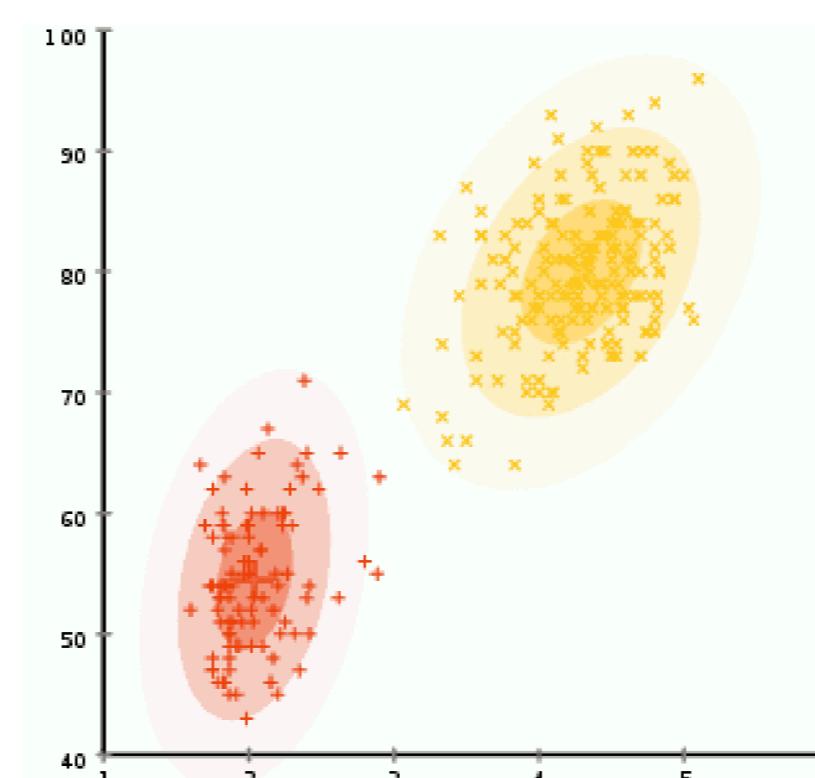
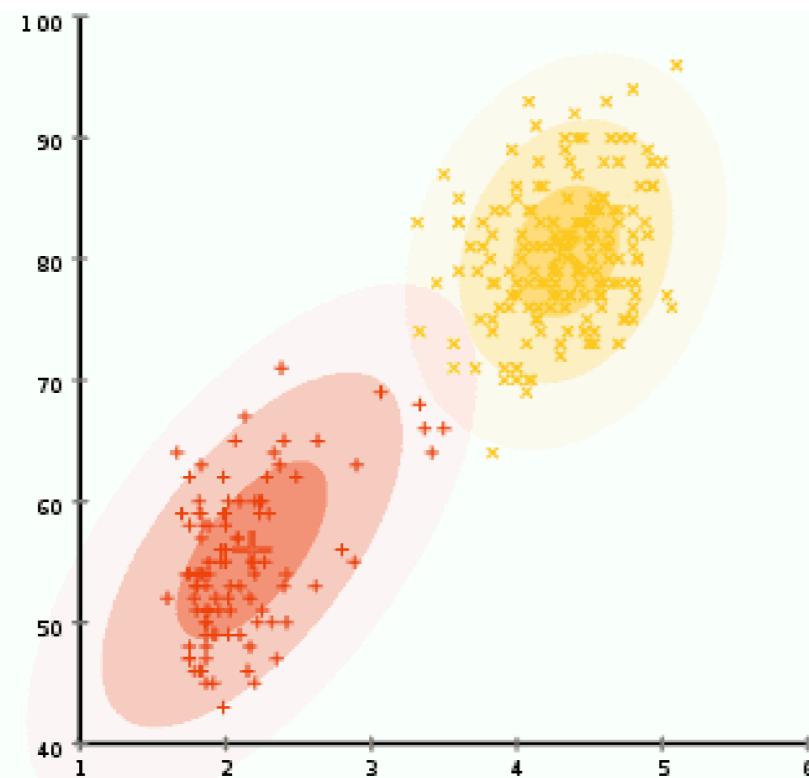
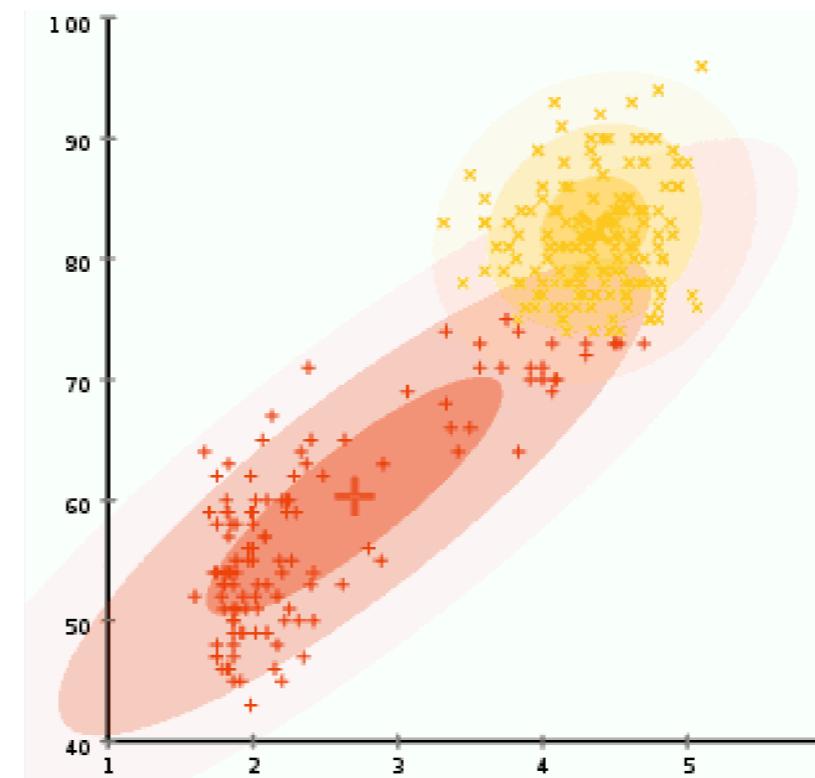
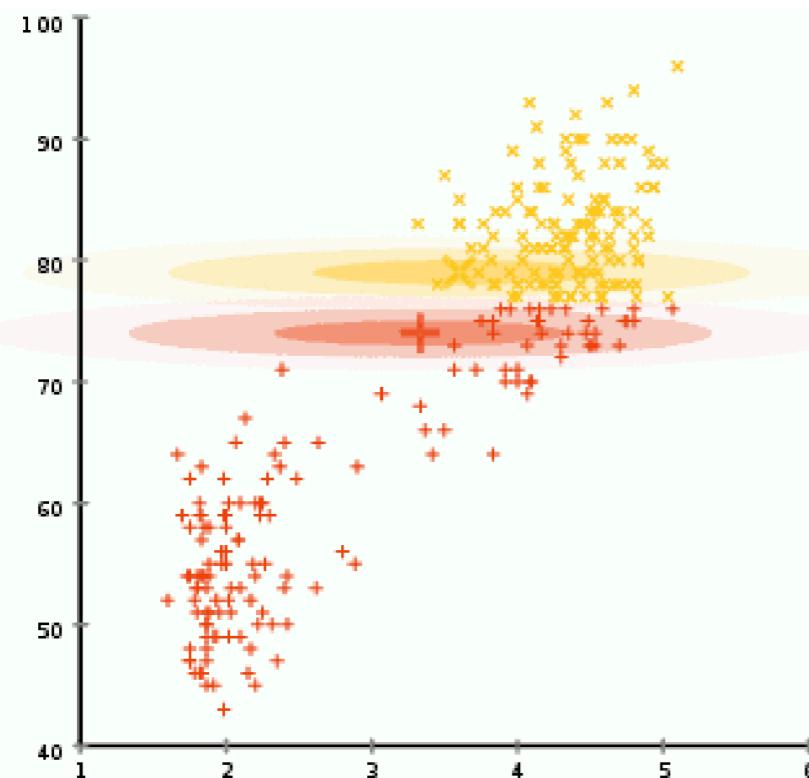
$$p_k(x) = \frac{1}{\sqrt{2\pi(\hat{\sigma}_k^{(t)})^2}} \exp\left(-\frac{(x - \hat{\mu}_k^{(t)})^2}{2(\hat{\sigma}_k^{(t)})^2}\right)$$

M-step: **for** $k = 1, \dots, K$, $\hat{\mu}_k^{(t+1)} \leftarrow \frac{\sum_{i=1}^n \hat{\eta}_{ik}^{(t+1)} X_i}{\sum_{i=1}^n \hat{\eta}_{ik}^{(t+1)}}$

$$(\hat{\sigma}_k^{(t+1)})^2 \leftarrow \frac{\sum_{i=1}^n \hat{\eta}_{ik}^{(t+1)} (X_i - \hat{\mu}_k^{(t+1)})^2}{\sum_{i=1}^n \hat{\eta}_{ik}^{(t+1)}}$$

$$\hat{\eta}_k^{(t+1)} \leftarrow \frac{1}{n} \sum_{i=1}^n \hat{\eta}_{ik}^{(t+1)}$$

EM Algorithm (Example)



General EM Algorithm

EM aims to **maximize the likelihood** of the mixture model

Flip a coin: $\mathbb{P}(Y = 1) = \eta$ and $\mathbb{P}(Y = 2) = 1 - \eta$

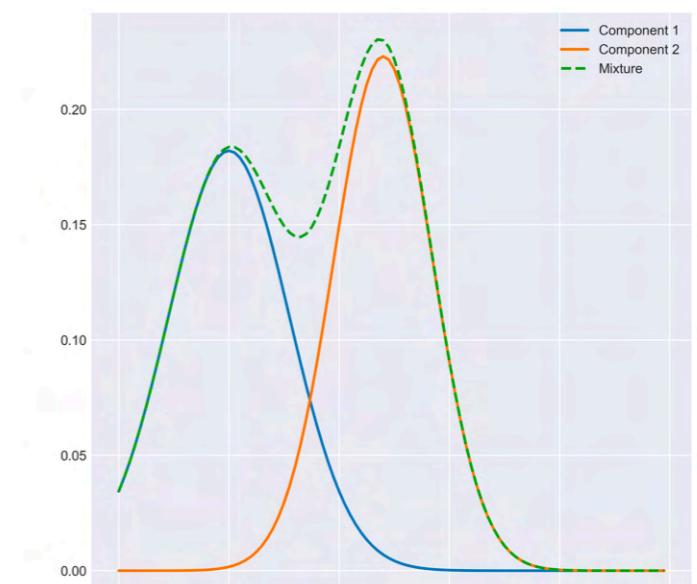
Generate X: $X|Y = 1 \sim N(\mu_1, 1), X|Y = 2 \sim N(\mu_2, 1)$

So the likelihood function $\ell(\theta)$ with $\theta = (\eta, \mu_1, \mu_2)$ is

$$\begin{aligned}\ell(\theta) &= \sum_{i=1}^n \log p_\theta(X_i) \\ &= \sum_{i=1}^n \log(\eta p_1(X_i) + (1 - \eta)p_2(X_i)) \\ &= \sum_{i=1}^n \log\left(\eta \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(X_i - \mu_1)^2}{2}\right) + (1 - \eta) \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(X_i - \mu_2)^2}{2}\right)\right)\end{aligned}$$

↑

Nonconvex and “log-sum” structure



Generic Geometry of EM

Nonconvex and “log-sum” structure of the likelihood

$$\ell(\theta) = \sum_{i=1}^n \log \left(\eta \frac{1}{\sqrt{2\pi}} \exp \left(- (X_i - \mu_1)^2 / 2 \right) + (1 - \eta) \frac{1}{\sqrt{2\pi}} \exp \left(- (X_i - \mu_2)^2 / 2 \right) \right)$$

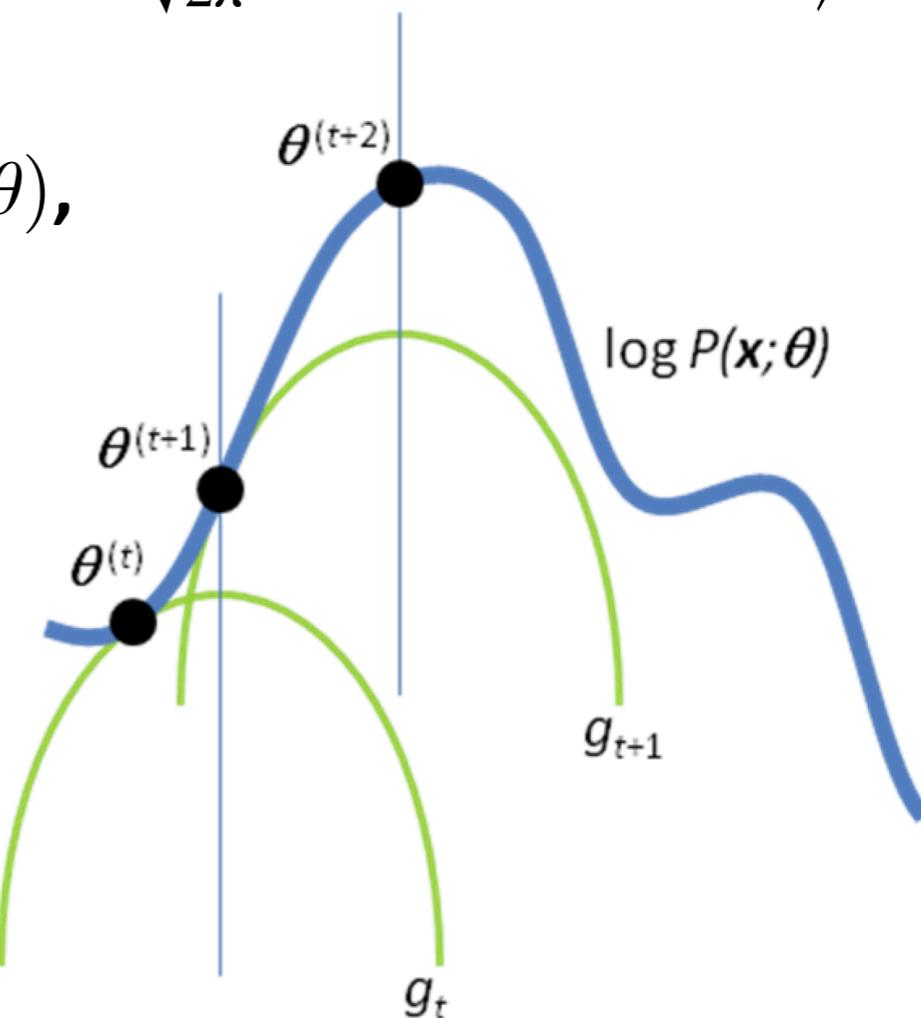
Key Idea of EM: To maximize complex $\ell(\theta)$,
we replace it locally by simple functions

For current $\theta^{(t)}$, find $g_t(\theta)$ s.t.

(1) $g_t(\theta) \leq \ell(\theta)$ (Lower bound)

(2) $g_t(\theta^{(t)}) = \ell(\theta^{(t)})$ (Tight at $\theta^{(t)}$)

(3) $\max_{\theta} g_t(\theta)$ is easy



repeat the above strategy until convergence



Question: How to find such $g_t(\theta)$?

Generic Geometry of EM

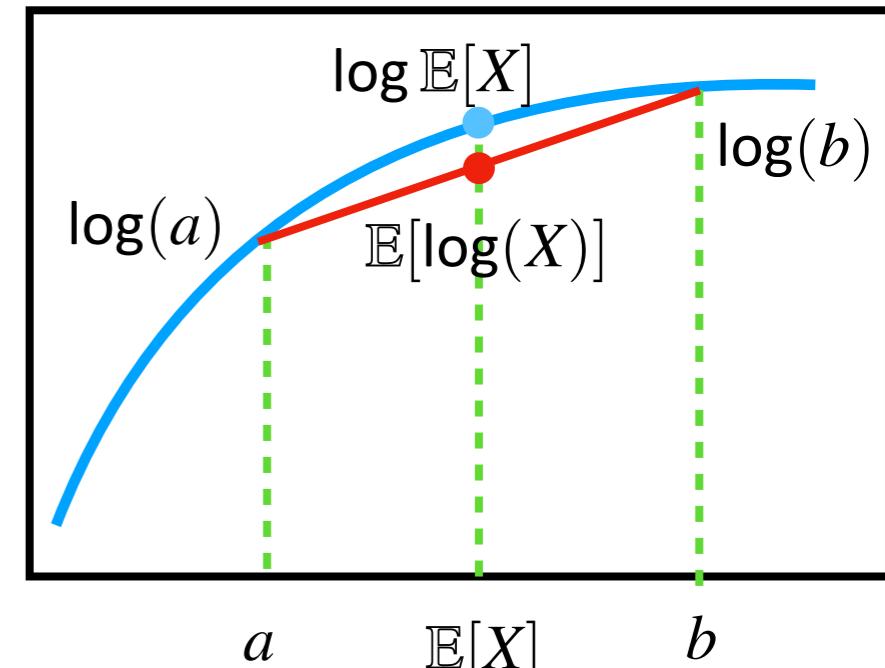
For current $\theta^{(t)}$, find $g_t(\theta)$ s.t. $g_t(\theta) \leq \ell(\theta)$ (Lower bound)

Answer: Jensen's inequality $\log \mathbb{E}[X] \geq \mathbb{E}[\log X]$

$$\begin{aligned}\ell(\theta) &= \sum_{i=1}^n \log p_\theta(X_i) \\ &= \sum_{i=1}^n \log \left[\sum_{k=1}^K p_\theta(X_i, Y_i = k) \right] \\ &= \sum_{i=1}^n \log \left[\sum_{k=1}^K \underbrace{\mathbb{P}_{\theta^{(t)}}(Y_i = k | X_i)}_{\eta_{ik}} \frac{p_\theta(X_i, Y_i = k)}{\mathbb{P}_{\theta^{(t)}}(Y_i = k | X_i)} \right]\end{aligned}$$

“log-sum”

η_{ik} is a probability dist. on $\{1, \dots, K\}$, $\eta_{ik} \geq 0$ and $\sum_{k=1}^K \eta_{ik} = 1$



$$\ell(\theta) = \sum_{i=1}^n \log \left[\sum_{k=1}^K \underbrace{\eta_{ik} \frac{p_\theta(X_i, Y_i = k)}{\eta_{ik}}}_{\mathbb{E}[p_\theta(X_i, Y_i = k) / \eta_{ik}]} \right] \geq \sum_{i=1}^n \underbrace{\sum_{k=1}^K \eta_{ik} \log \left[\frac{p_\theta(X_i, Y_i = k)}{\eta_{ik}} \right]}_{\mathbb{E}[\log(p_\theta(X_i, Y_i = k) / \eta_{ik})]} := g_t(\theta)$$

↑
Jensen

Generic Geometry of EM

For current $\theta^{(t)}$, find $g_t(\theta)$ s.t.

(1) $g_t(\theta) \leq \ell(\theta)$ (Lower bound)



(2) $g_t(\theta^{(t)}) = \ell(\theta^{(t)})$ (Tight at $\theta^{(t)}$)



(3) $\max_{\theta} g_t(\theta)$ is easy

$$\begin{aligned} g_t(\theta^{(t)}) &= \sum_{i=1}^n \sum_{k=1}^K \eta_{ik} \log \left[\frac{p_{\theta^{(t)}}(X_i, Y_i = k)}{\eta_{ik}} \right] \\ &= \sum_{i=1}^n \sum_{k=1}^K \eta_{ik} \log \left[\frac{p_{\theta^{(t)}}(Y_i = k | X_i) p_{\theta^{(t)}}(X_i)}{\eta_{ik}} \right] \\ &= \sum_{i=1}^n \sum_{k=1}^K \eta_{ik} \log p_{\theta^{(t)}}(X_i) = \sum_{i=1}^n \log p_{\theta^{(t)}}(X_i) = \ell(\theta^{(t)}) \end{aligned}$$

Generic Geometry of EM

For current $\theta^{(t)}$, find $g_t(\theta)$ s.t. (3) $\max_{\theta} g_t(\theta)$ is easy ✓

$$\operatorname{argmax}_{\theta} g_t(\theta) = \operatorname{argmax}_{\theta} \sum_{i=1}^n \sum_{k=1}^K \eta_{ik} \log \left[\frac{p_{\theta}(X_i, Y_i = k)}{\eta_{ik}} \right]$$

$$= \operatorname{argmax}_{\theta} \sum_{i=1}^n \sum_{k=1}^K \eta_{ik} (\log p_{\theta}(X_i, Y_i = k) - \log \eta_{ik})$$

↑
not relevant to θ

$$= \operatorname{argmax}_{\theta} \sum_{i=1}^n \sum_{k=1}^K \eta_{ik} \log p_{\theta}(X_i, Y_i = k)$$

$$= \operatorname{argmax}_{\theta} \sum_{i=1}^n \sum_{k=1}^K \eta_{ik} \log(p_{\theta_k}(X_i|Y_i = k)\eta_k) \quad (\theta_k: \text{parameters like } \mu_k, \sigma_k)$$

$$= \operatorname{argmax}_{\theta} \sum_{i=1}^n \sum_{k=1}^K \eta_{ik} \log p_{\theta_k}(X_i|Y_i = k) + \sum_{i=1}^n \sum_{k=1}^K \eta_{ik} \log \eta_k$$

Generic Geometry of EM

For current $\theta^{(t)}$, find $g_t(\theta)$ s.t. (3) $\max_{\theta} g_t(\theta)$ is easy ✓

$$\operatorname{argmax}_{\theta} g_t(\theta) = \operatorname{argmax}_{\theta} \sum_{i=1}^n \sum_{k=1}^K \eta_{ik} \log p_{\theta_k}(X_i | Y_i = k) + \sum_{i=1}^n \sum_{k=1}^K \eta_{ik} \log \eta_k$$

Therefore: $\theta_k^{(t+1)} \leftarrow \operatorname{argmax}_{\theta_k} \sum_{i=1}^n \sum_{k=1}^K \eta_{ik} \log p_{\theta_k}(X_i | Y_i = k)$

$$\eta_k^{(t+1)} \leftarrow \operatorname{argmax}_{\eta_1, \dots, \eta_K} \sum_{i=1}^n \sum_{k=1}^K \eta_{ik} \log \eta_k$$

$$\text{s.t. } \sum_{k=1}^K \eta_k = 1$$

We have closed form solution for the second

$$\eta_k^{(t+1)} \leftarrow \frac{1}{n} \sum_{i=1}^n \eta_{ik}$$

Generic Geometry of EM

The EM Algorithm: **Initialization.** Randomly initialize $\theta^{(0)}$

For $t = 0, 1, 2, \dots$

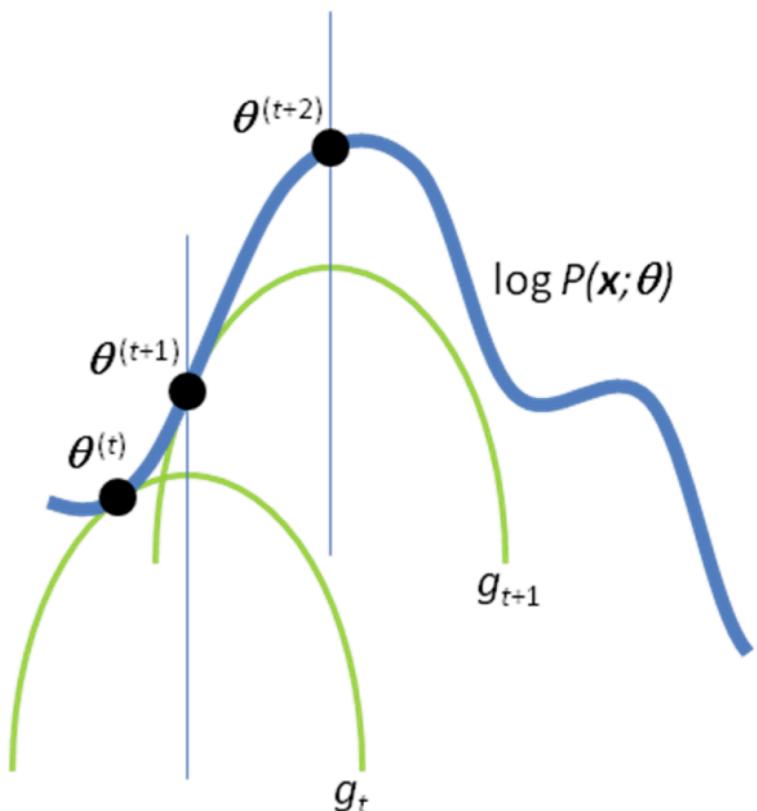
E-step: Construct $g_t(\theta)$

Key: Evaluate $\eta_{ik} = \mathbb{P}(Y_i = k|X_i) = \mathbb{E}[\mathbb{I}(Y_i = k)|X_i]$ (**Expectation**)

M-step:

$$\theta^{(t+1)} \leftarrow \operatorname{argmax}_{\theta} g_t(\theta)$$

Iterate E&M steps until converges



Generic Geometry of EM

The EM Algorithm (Formal)

Initialization. Randomly initialize $\theta^{(0)} = (\eta_1^{(0)}, \dots, \eta_K^{(0)}, \theta_1^{(0)}, \dots, \theta_k^{(0)})$

For $t = 0, 1, 2, \dots$

E-step:

$$\eta_{ik}^{(t+1)} := \mathbb{P}_{\theta^{(t)}}(Y_i = k | X_i) = \frac{p_{\theta^{(t)}}(X_i | Y_i = k) \eta_k^{(t)}}{\sum_{k=1}^K p_{\theta^{(t)}}(X_i | Y_i = k) \eta_k^{(t)}}$$

M-step:

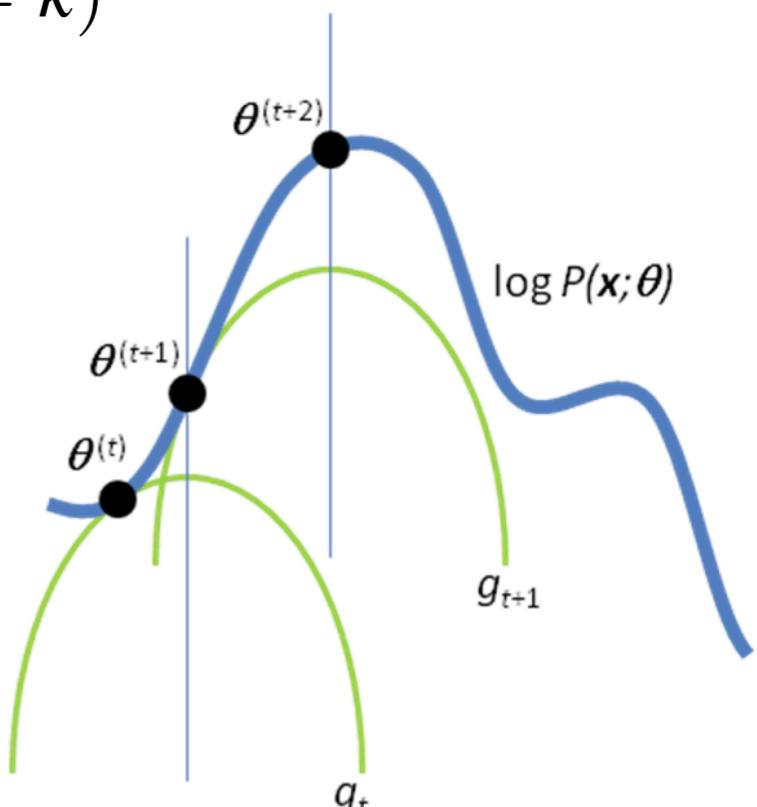
$$\theta_k^{(t+1)} = \operatorname{argmax}_{\theta_k} \sum_{i=1}^n \sum_{k=1}^K \eta_{ik}^{(t+1)} \log p_{\theta_k}(X_i | Y_i = k)$$

$$\eta_k^{(t+1)} = \frac{1}{n} \sum_{i=1}^n \eta_{ik}^{(t+1)}$$

Iterate E&M steps until converges



Question: Does it match EM for Gaussian?



EM Algorithm

Expectation-Maximization (K mixture Gaussian):

Initialization. Randomly initialize $\hat{\mu}_k^{(0)}, \hat{\sigma}_k^{(0)}, \hat{\eta}^{(0)}$ for $k = 1, \dots, K$

For $t = 0, 1, 2, \dots$

E-step: **for** $k = 1, \dots, K$, $\hat{\eta}_{ik}^{(t+1)} \leftarrow \frac{p_k(X_i)\hat{\eta}_k^{(t)}}{\sum_{k=1}^K p_k(X_i)\hat{\eta}_k^{(t)}}$

$$p_k(x) = \frac{1}{\sqrt{2\pi(\hat{\sigma}_k^{(t)})^2}} \exp\left(-\frac{(x - \hat{\mu}_k^{(t)})^2}{2(\hat{\sigma}_k^{(t)})^2}\right)$$

M-step: **for** $k = 1, \dots, K$, $\theta_k^{(t+1)} = \operatorname{argmax}_{\theta_k} \sum_{i=1}^n \sum_{k=1}^K \hat{\eta}_{ik}^{(t+1)} \log p_{\theta_k}(X_i|Y_i=k)$

$$\hat{\mu}_k^{(t+1)} \leftarrow \frac{\sum_{i=1}^n \hat{\eta}_{ik}^{(t+1)} X_i}{\sum_{i=1}^n \hat{\eta}_{ik}^{(t+1)}} \quad (\hat{\sigma}_k^{(t+1)})^2 \leftarrow \frac{\sum_{i=1}^n \hat{\eta}_{ik}^{(t+1)} (X_i - \hat{\mu}_k^{(t+1)})^2}{\sum_{i=1}^n \hat{\eta}_{ik}^{(t+1)}}$$

$$\hat{\eta}_k^{(t+1)} \leftarrow \frac{1}{n} \sum_{i=1}^n \hat{\eta}_{ik}^{(t+1)}$$

Connection between EM and K-Means

EM Algorithm for $X|Y = k \sim N(\mu_k, \sigma)$

E-step:

$$\eta_{ik}^{(t+1)} = \frac{p_{\theta^{(t)}}(X_i|Y_i = k)\eta_k^{(t)}}{\sum_{k=1}^K p_{\theta^{(t)}}(X_i|Y_i = k)\eta_k^{(t)}} = \frac{\eta_k^{(t)} \exp(-\|X_i - \mu_k\|_2^2 / (2\sigma^2))}{\sum_{k=1}^K \eta_k^{(t)} \exp(-\|X_i - \mu_k\|_2^2 / (2\sigma^2))}$$

Assuming no ties, when $\sigma^2 \rightarrow 0$, we have

$$\eta_{ik} = \begin{cases} 1 & \text{if } \|X_i - \mu_k\|_2^2 < \|X_i - \mu_j\|_2^2 \text{ for all } k \neq j \\ 0 & \text{otherwise} \end{cases}$$

This is **hard assignment**: K-means is the limit algorithm of EM

In comparison, EM has **soft assignment**

Think about $5/(5+3(y/x))$ as $x, y \rightarrow 0$ it depends on $y/x \rightarrow 0$, $5/(5+3(y/x)) \rightarrow 1$
 $y/x \rightarrow \infty$, $5/(5+3(y/x)) \rightarrow 0$

General EM for Latent Model

We wish to fit the parameters of a model $p_\theta(x, z)$

But we only observe the data X_1, \dots, X_n

Not observe latent variables: Z_1, \dots, Z_n (mixture, missing data, etc)

Example: the mixture model

Flip a coin / Toss a dice: $\mathbb{P}(Y = k) = \eta_k$

Generate observations: $X|Y = k \sim p_k(x)$

So the latent model: $p(x, z) = \sum_{i=1}^K p_k(x)\mathbb{I}(z = k)$

General EM for Latent Model

The EM Algorithm for $p_\theta(x, z)$ with discrete z

Initialization. Randomly initialize $\theta^{(0)}$

For $t = 0, 1, 2, \dots$

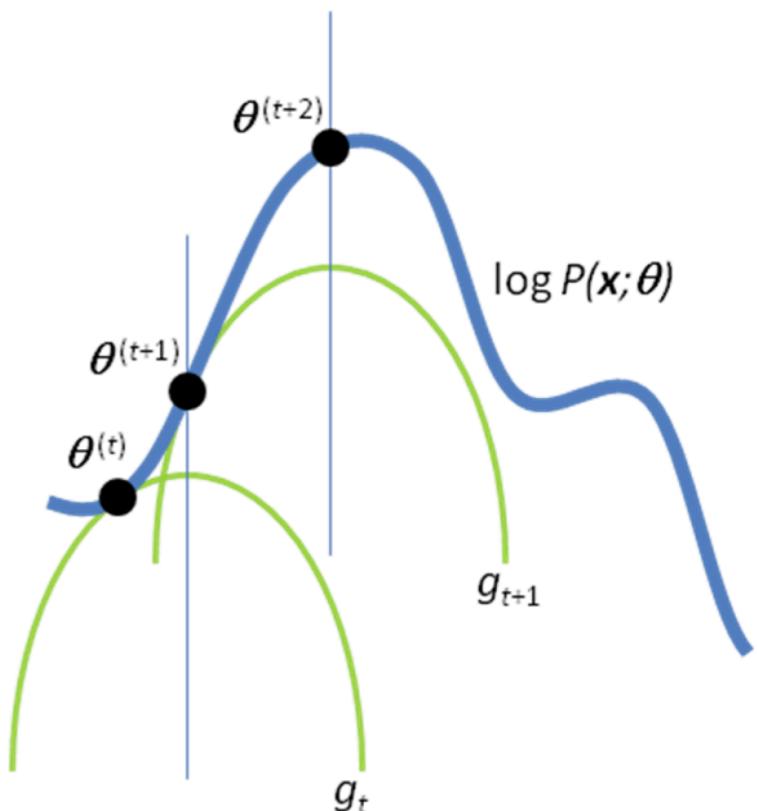
E-step:

$$\eta_i^{(t+1)}(z) = \mathbb{P}_{\theta^{(t)}}(z|X_i)$$

M-step:

$$\theta^{(t+1)} = \operatorname{argmax}_\theta \sum_{i=1}^n \sum_z \eta_i^{(t+1)}(z) \log \frac{p_\theta(X_i, z)}{\eta_i^{(t+1)}(z)}$$

Iterate E&M steps until converges



General EM for Latent Model

The EM Algorithm for $p_{\theta}(x, z)$ with continuous z

Initialization. Randomly initialize $\theta^{(0)}$

For $t = 0, 1, 2, \dots$

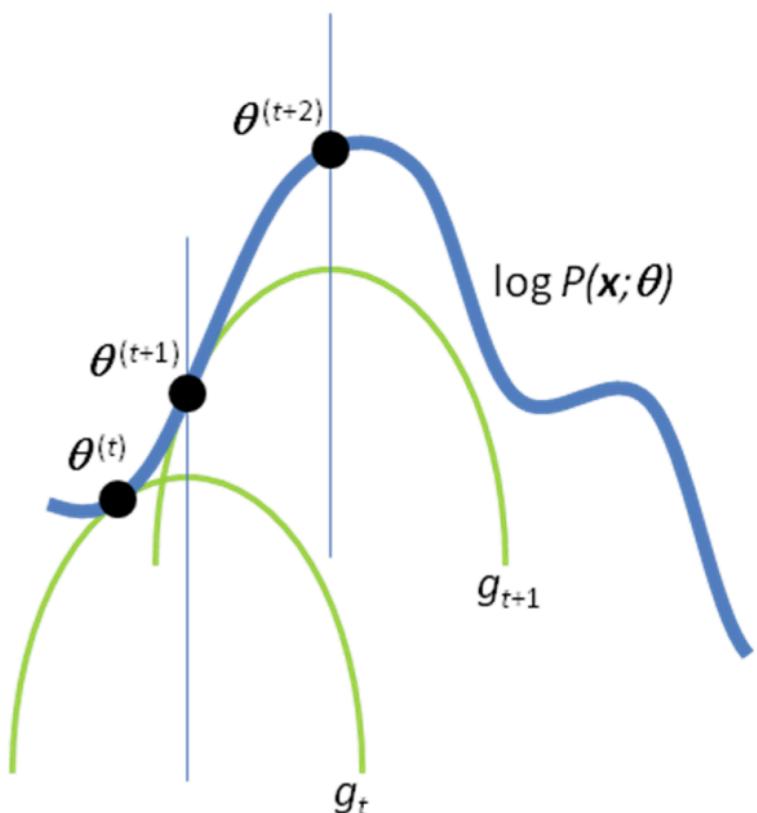
E-step:

$$\eta_i^{(t+1)}(z) = p_{\theta^{(t)}}(z|X_i)$$

M-step:

$$\theta^{(t+1)} = \operatorname{argmax}_{\theta} \sum_{i=1}^n \int \eta_i^{(t+1)}(z) \log \frac{p_{\theta}(X_i, z)}{\eta_i^{(t+1)}(z)} dz$$

Iterate E&M steps until converges



Summary of Clustering

- **Model-free methods**

- Center based: K-means**

- Hierarchical based: hierarchical clustering**

- How to select number of clusters**

- **Model-based methods**

- Mixture model**

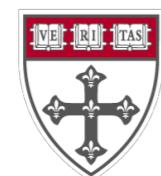
- EM algorithm for Gaussian mixture**

- EM algorithm for general mixture model**

- The generic geometry of EM**

Dimension Reduction: PCA

Junwei Lu



HARVARD
T.H. CHAN

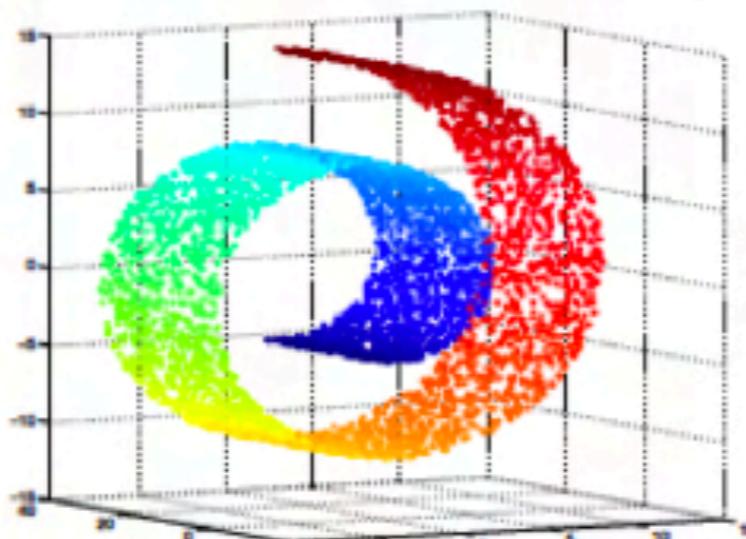
SCHOOL OF PUBLIC HEALTH
Department of Biostatistics



Curse of Dimensionality

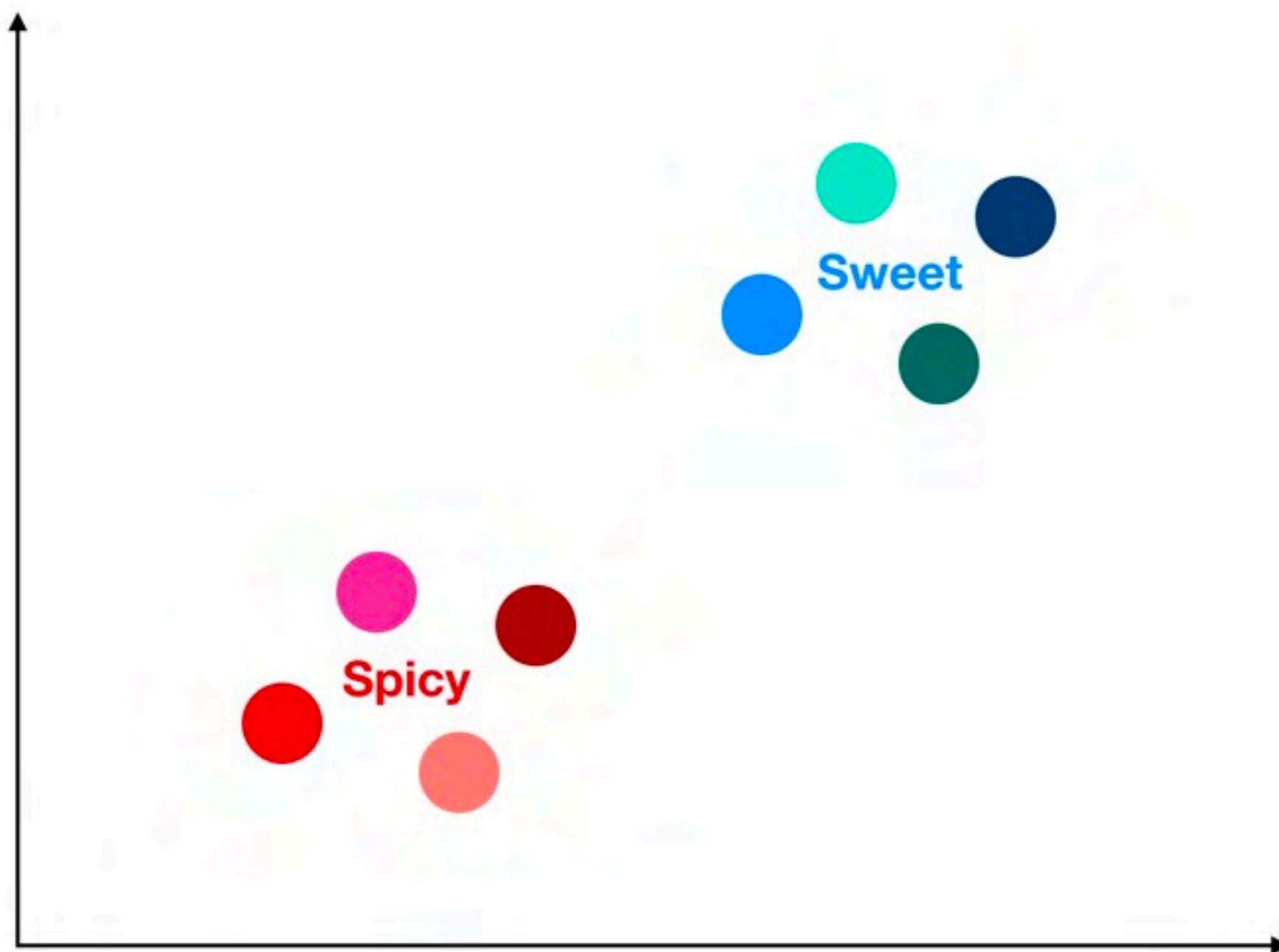
If the data dimension is too large

- Not interpretable/Hard to visualize
- Computation workload too high
- Most data is intrinsically low dimension



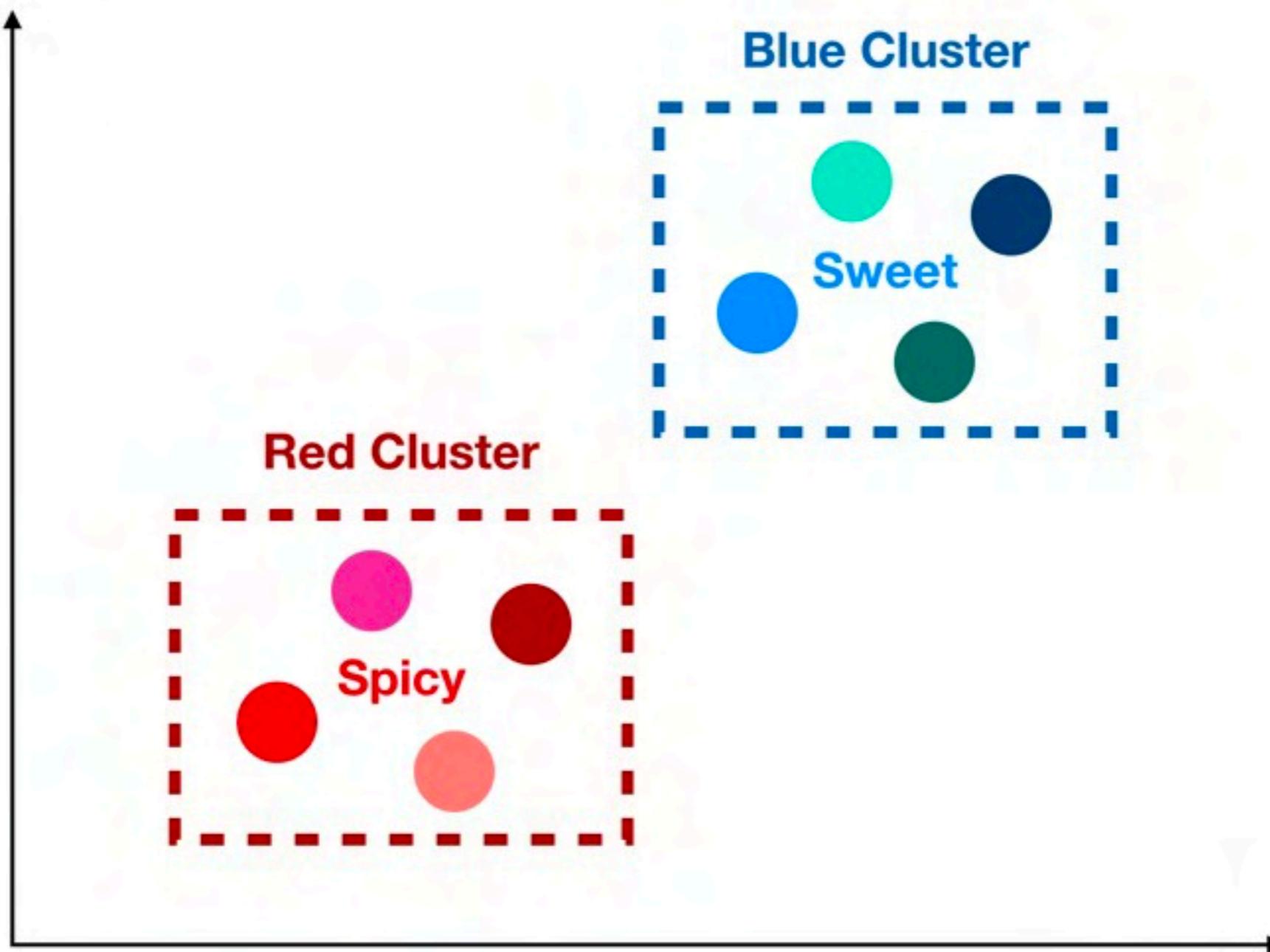
Curse of Dimensionality in Clustering

Imagine our dataset consists of the following 8 candies.



Curse of Dimensionality in Clustering

We can cluster by color like so:



Nice, thanks to our clustering we know that if we eat a reddish candy, it will be spicy; and if we eat a bluish candy, it will be sweet.

Curse of Dimensionality in Clustering

But actually it's not that simple to computer

	Reddish	Bluish
1	1	0
2	1	0
3	1	0
4	1	0
5	0	1
6	0	1
7	0	1
8	0	1

Computer only recognizes numbers. If data looks like above, we are lucky.

Curse of Dimensionality in Clustering

However, if the dataset is high dimensional

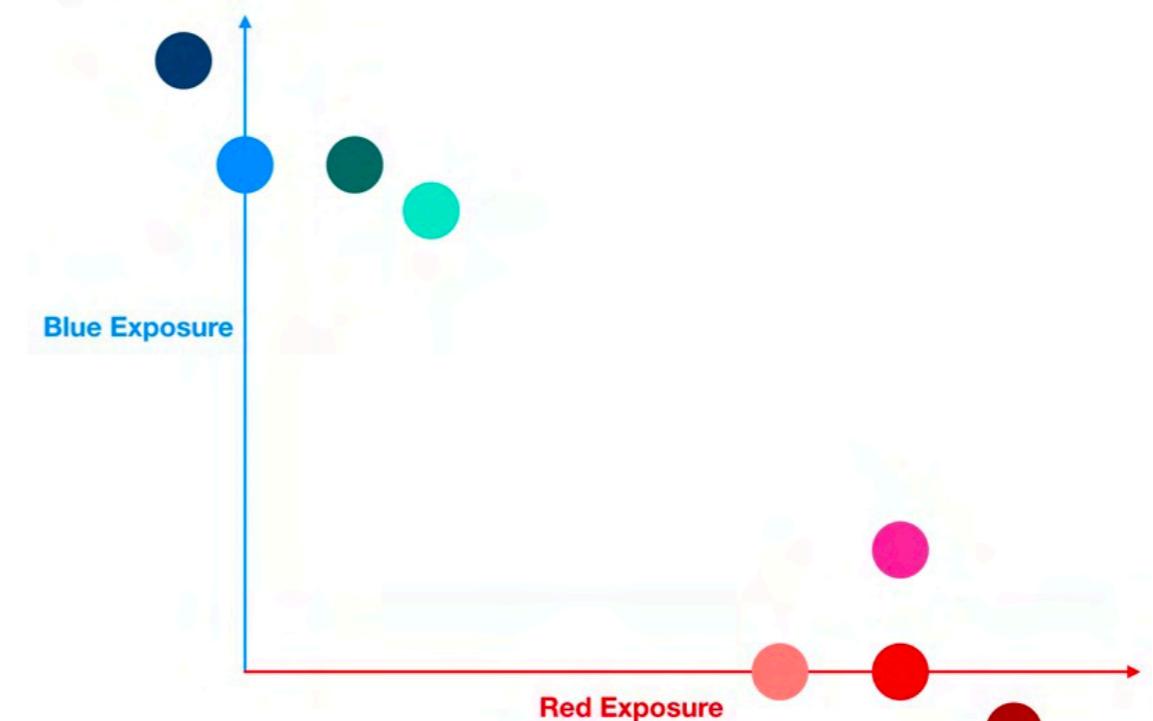
	Red	Maroon	Pink	Flamingo	Blue	Turquoise	Seaweed	Ocean
	1	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0
	0	0	1	0	0	0	0	0
	0	0	0	1	0	0	0	0
	0	0	0	0	1	0	0	0
	0	0	0	0	0	1	0	0
	0	0	0	0	0	0	1	0
	0	0	0	0	0	0	0	1

Recall the zip code in homework?

Curse of Dimensionality in Clustering

We need to reduce the dimension like this

	Red	Blue
Red	1.00	0
Maroon	1.20	-0.10
Pink	1.00	0.20
Flamingo	0.80	0
Blue	0	1.00
Turquoise	0.25	0.90
Seaweed	0.15	1.00
Ocean	-0.10	1.20

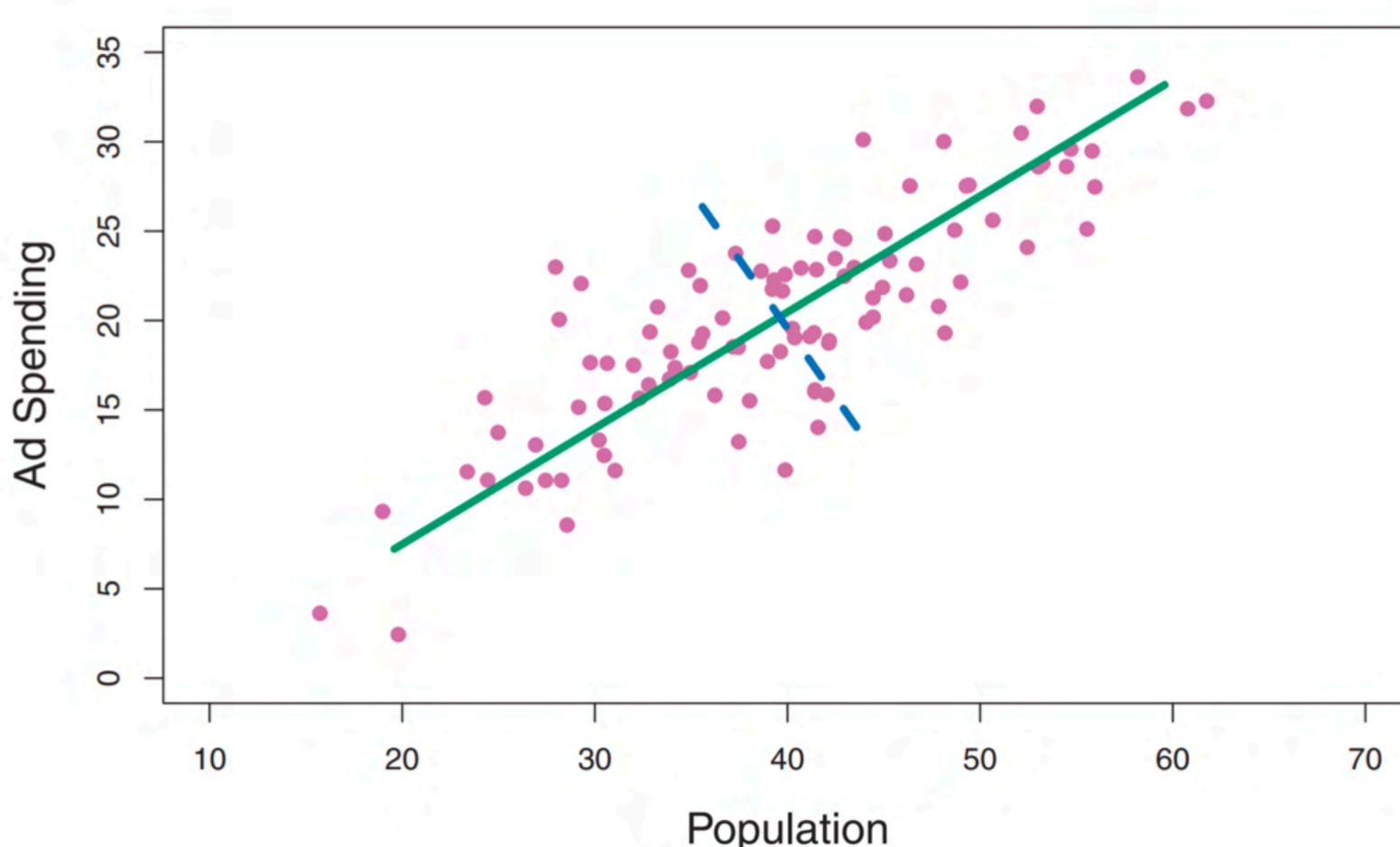


Principal Components Analysis

PCA is an **unsupervised** method for dimension reduction.

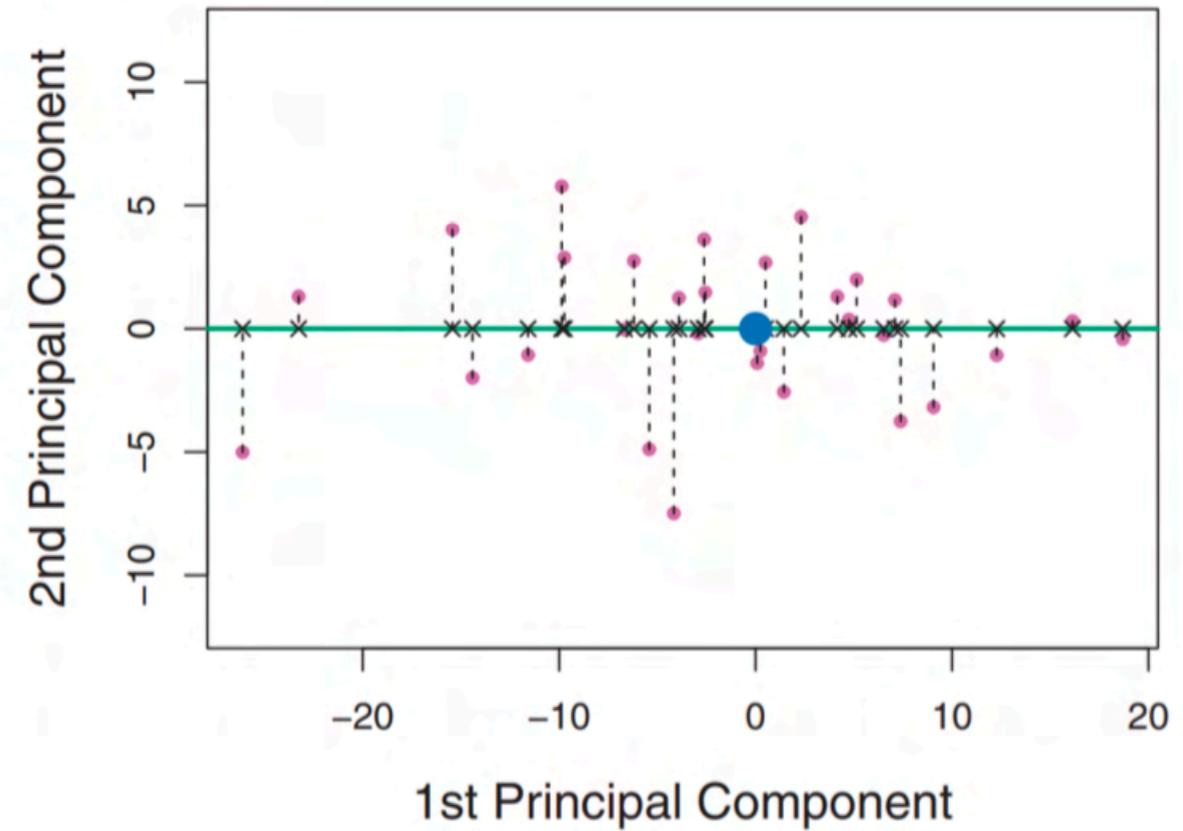
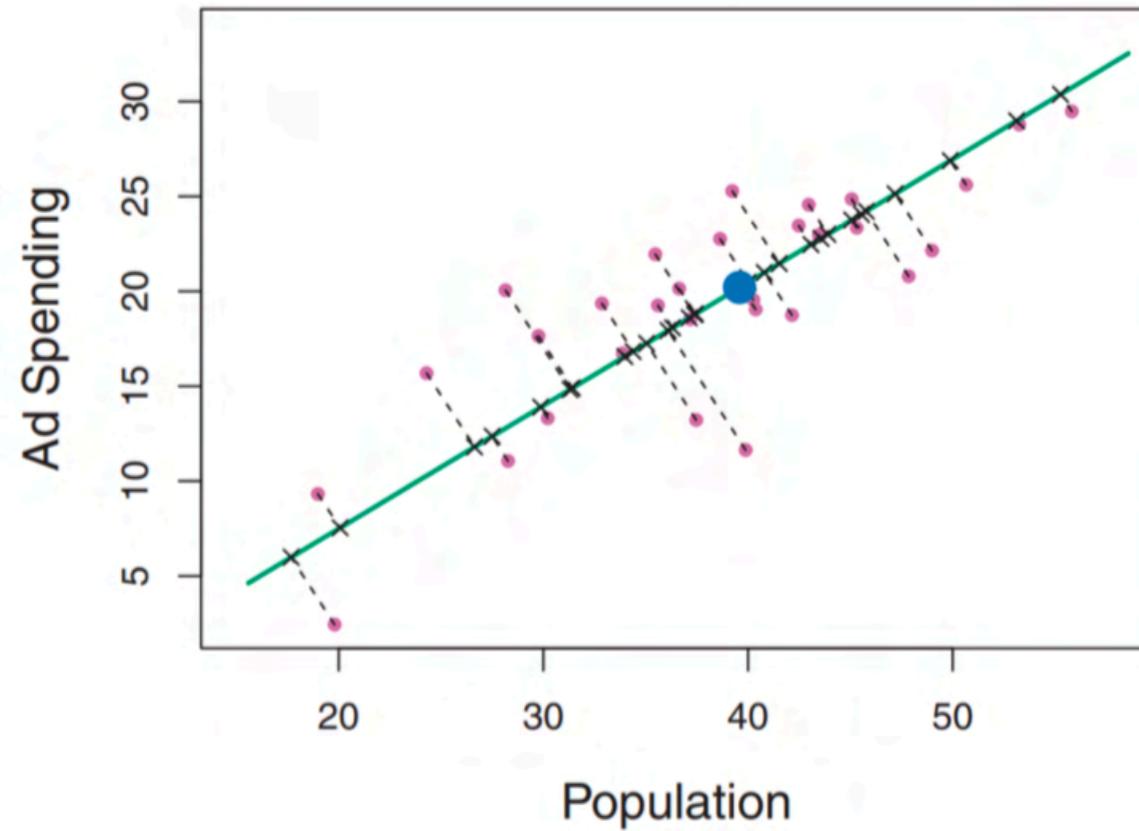
Basic idea: Find a low-dimensional representation that approximates the data as closely as possible in Euclidean distance.

Example: Population size versus Ad spending



Principal Components Analysis

Example: Population size versus Ad spending

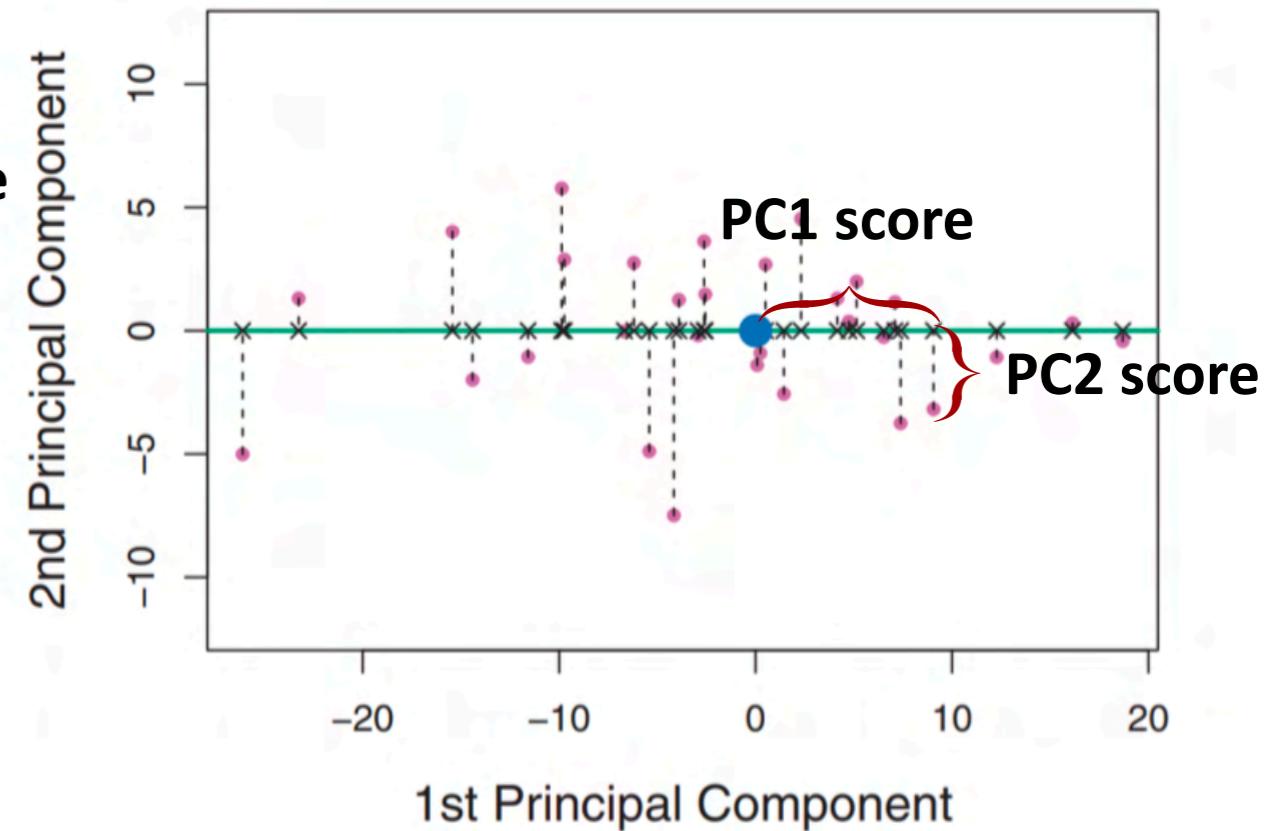
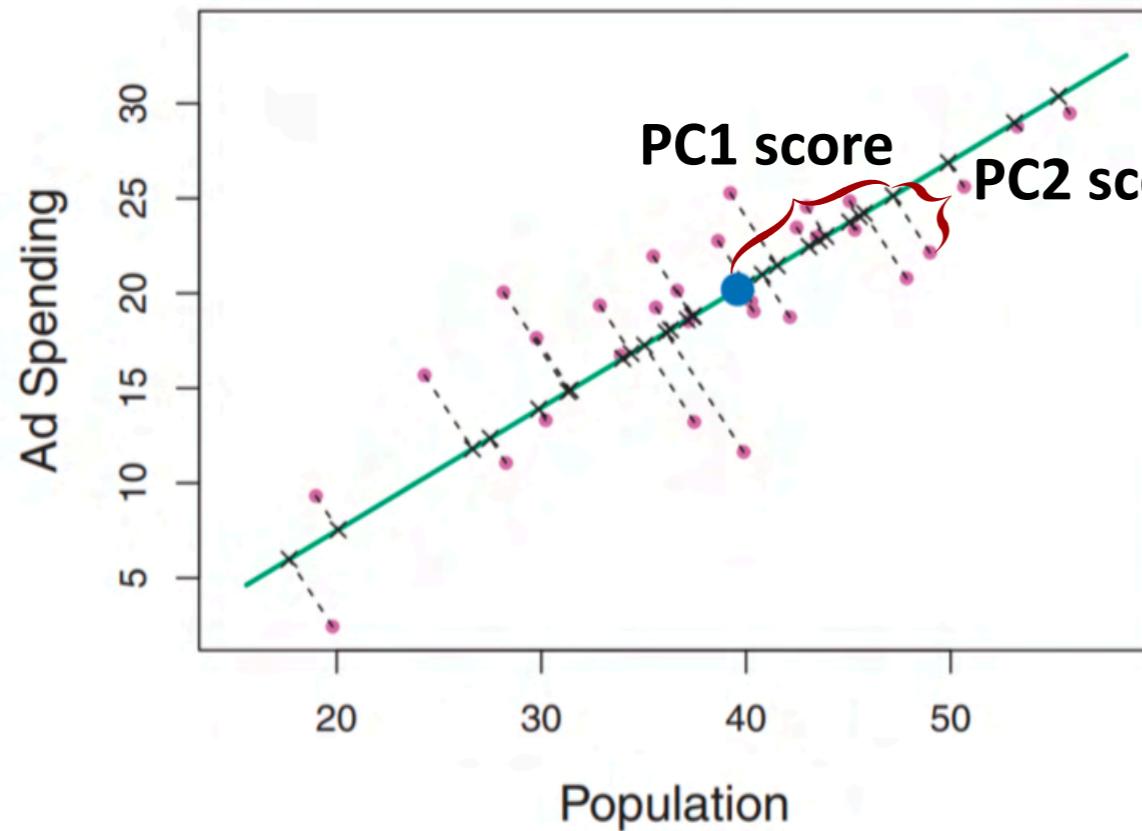


PC direction:

- *PC₁ direction* is the direction along which the data has the largest variance.
- *PC_m direction* is the direction along which the data has the largest variance, among all directions orthogonal to the first $m - 1$ PC directions.

Principal Components Analysis

Example: Population size versus Ad spending



PC score:

- PC_m score of a data point is its projection to the PC_m direction

Principal Components Analysis

PC1 score × PC1 direction is

the best 1-dimensional approximation to the data in terms of MSE.

$\sum_{m=1}^M \text{PC}_m \text{ score} \times \text{PC}_m \text{ direction}$ is

the best M -dimensional approximation to the data in terms of MSE.



Question: How to get PC direction and scores?

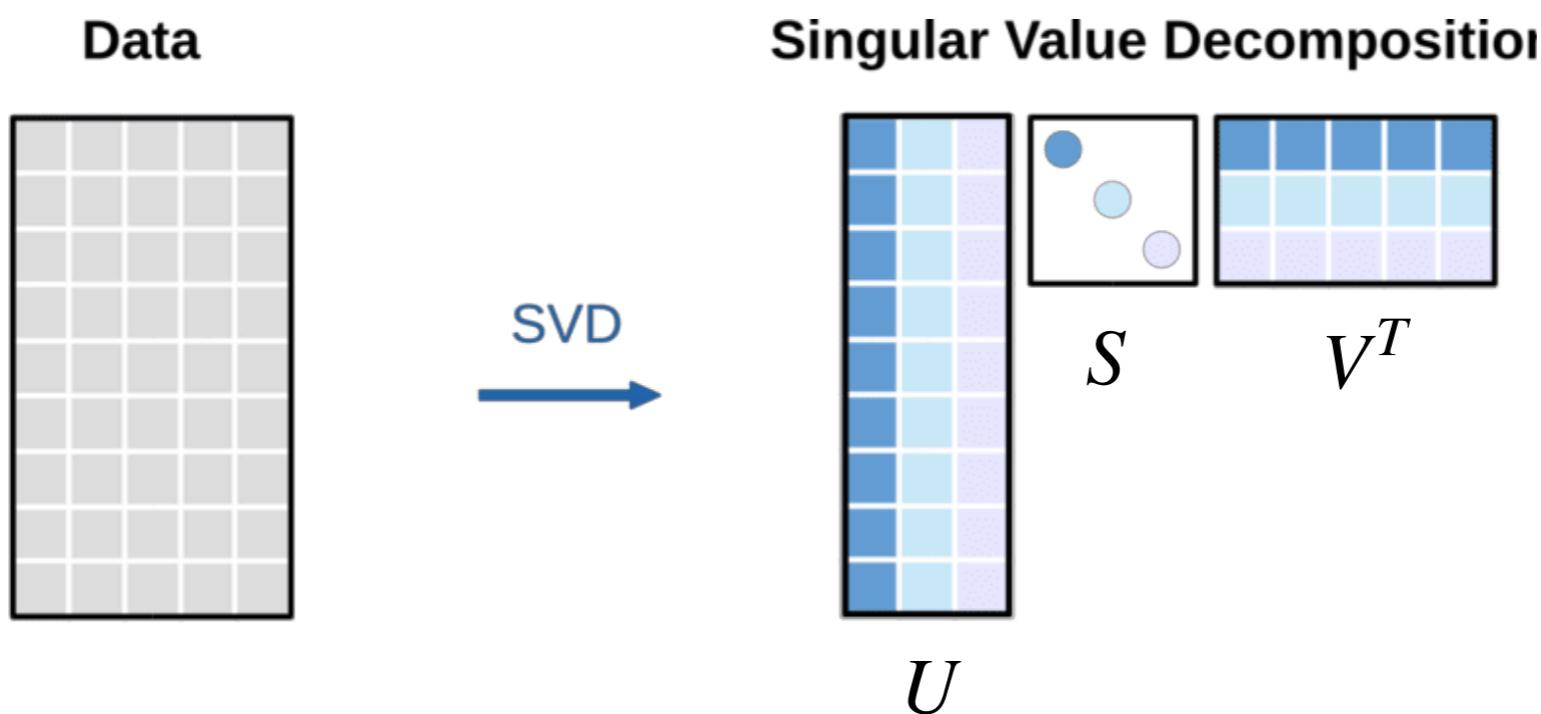
SVD method for computing PCA

SVD = Singular Value Decomposition.

Step 1. Center the data so that $\frac{1}{n} \sum_{i=1}^n x_{ij} = 0$ for each j .

Step 2. Put the data into an $n \times p$ matrix: $X = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$.

Step 3. Compute the SVD: $X = USV^T$.



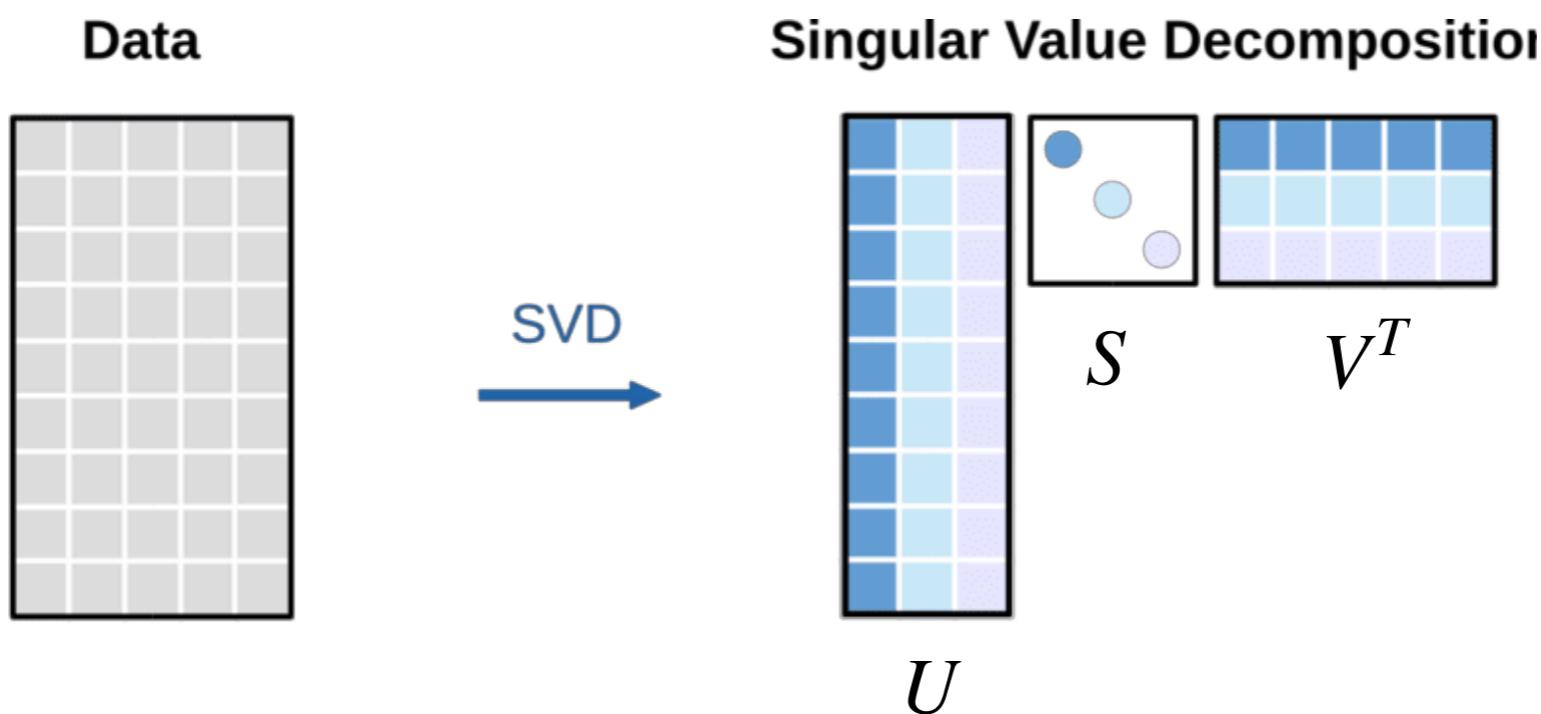
SVD method for computing PCA

SVD = Singular Value Decomposition.

Step 1. Center the data so that $\frac{1}{n} \sum_{i=1}^n x_{ij} = 0$ for each j .

Step 2. Put the data into an $n \times p$ matrix: $X = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$.

Step 3. Compute the SVD: $X = USV^T$.



SVD method for computing PCA

SVD = Singular Value Decomposition.

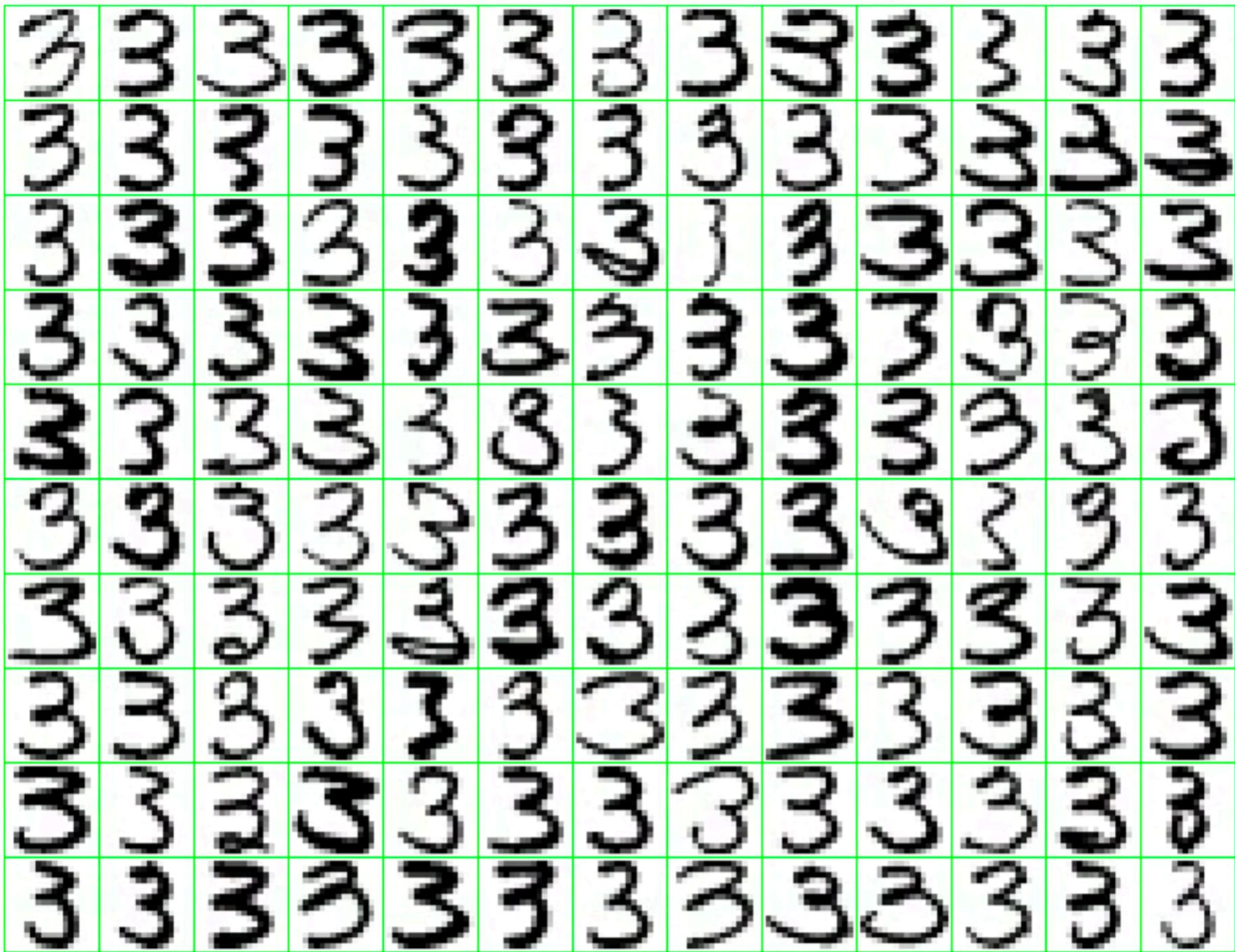
Step 1. Center the data so that $\frac{1}{n} \sum_{i=1}^n x_{ij} = 0$ for each j .

Step 2. Put the data into an $n \times p$ matrix: $X = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$.

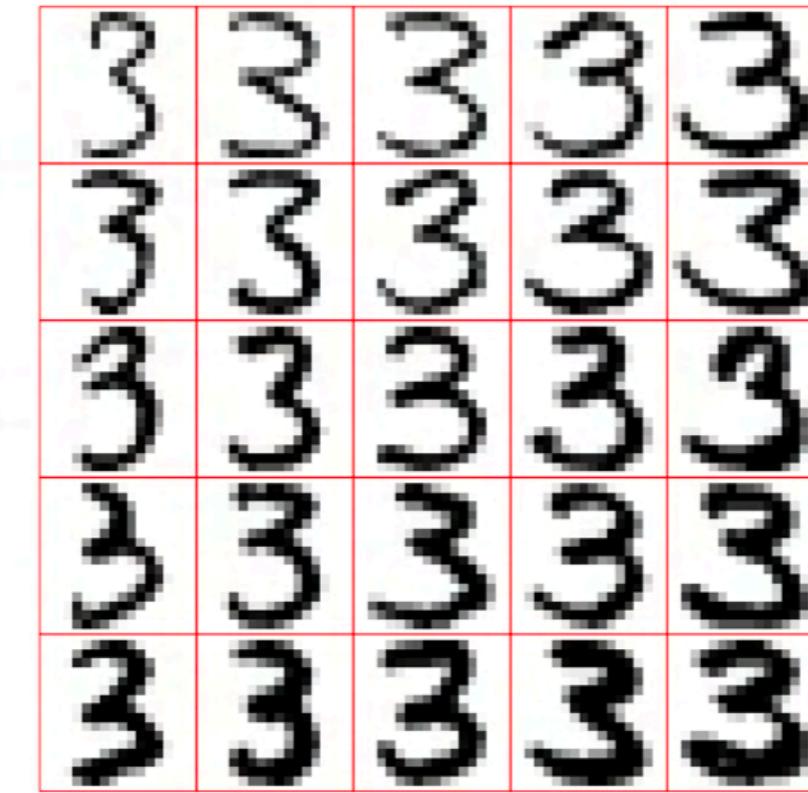
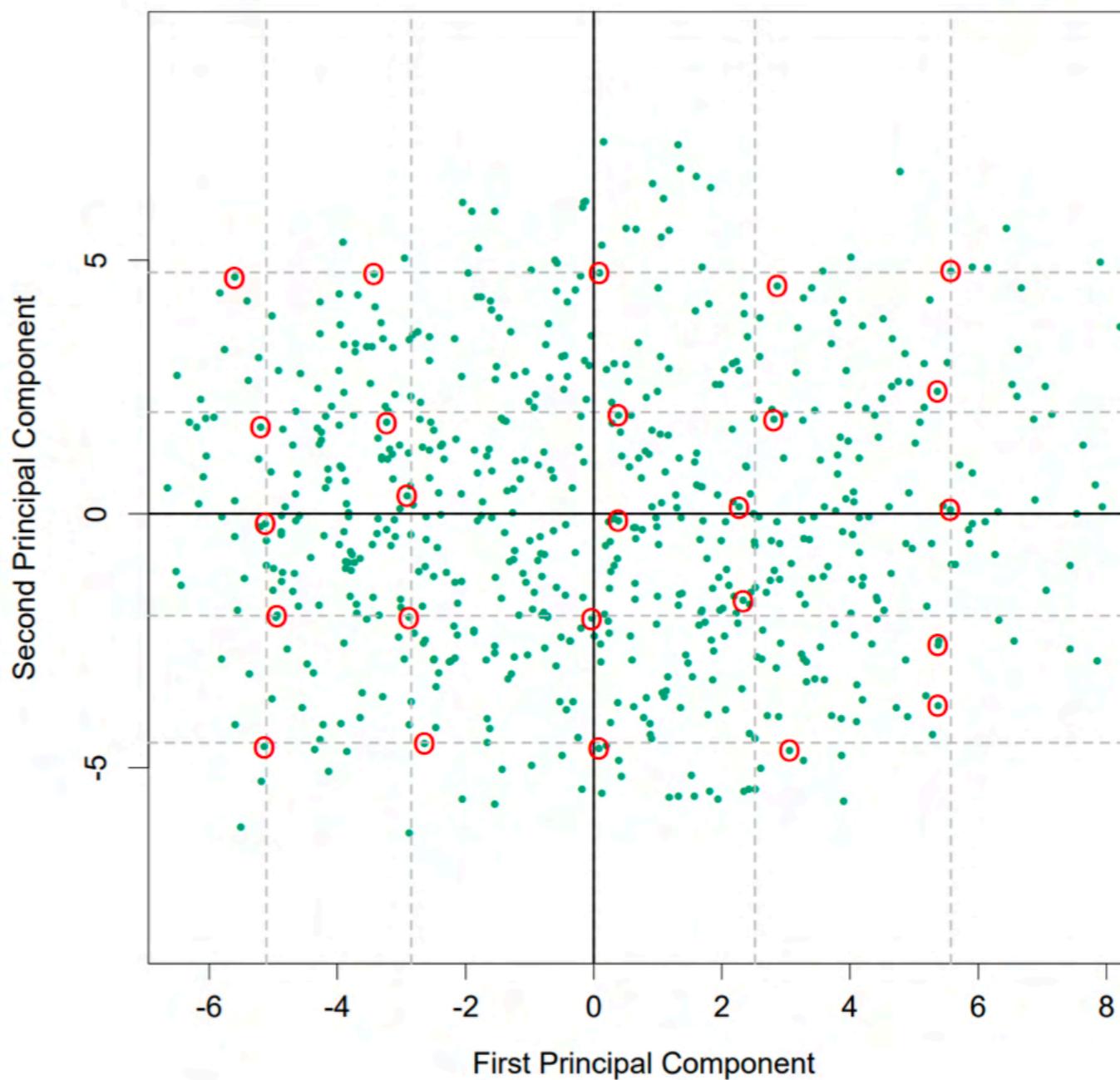
Step 3. Compute the SVD: $X = USV^T$.

- PC m direction is the m th column of V .
- PC m scale is $\frac{1}{\sqrt{n-1}} S_{mm}$.
- PC score vector (i.e., PC1-PC p scores) for x_i is $V^T x_i$.

PCA Example: Hand Writing Digits



PCA Example: Hand Writing Digits



PCA can be used to make a low-dimensional approximation to each image.

PCA Example: Hand Writing Digits

PCA can be used to make a low-dimensional approximation to each image.

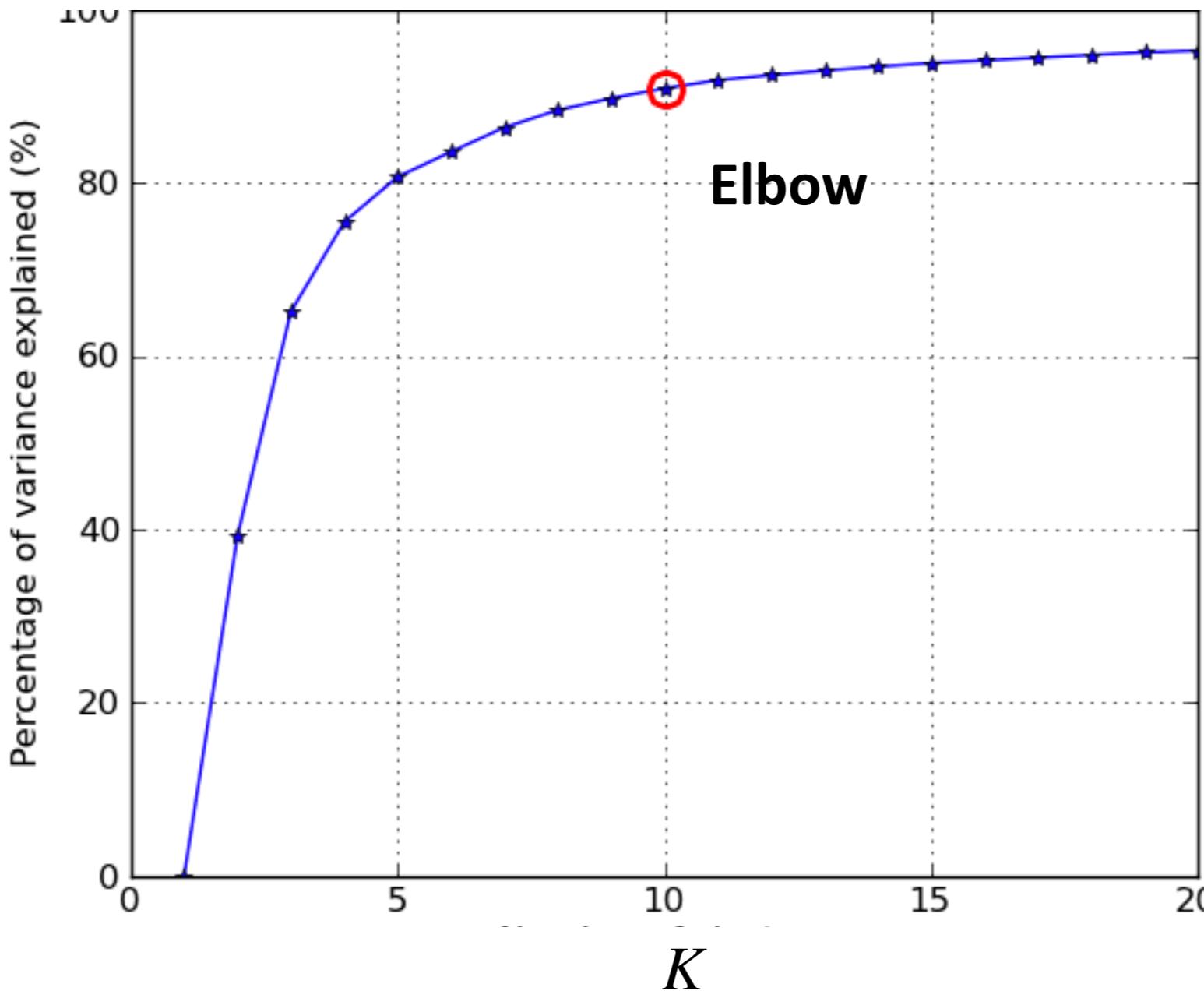
$$\text{PC approx} = \text{sample mean} + \text{score1} \times \text{dir1} + \text{score2} \times \text{dir2}$$

$$= \boxed{\text{3}} + \text{score1} \times \boxed{\text{3}} + \text{score2} \times \boxed{\text{3}}$$

In this example, each PC direction is image itself

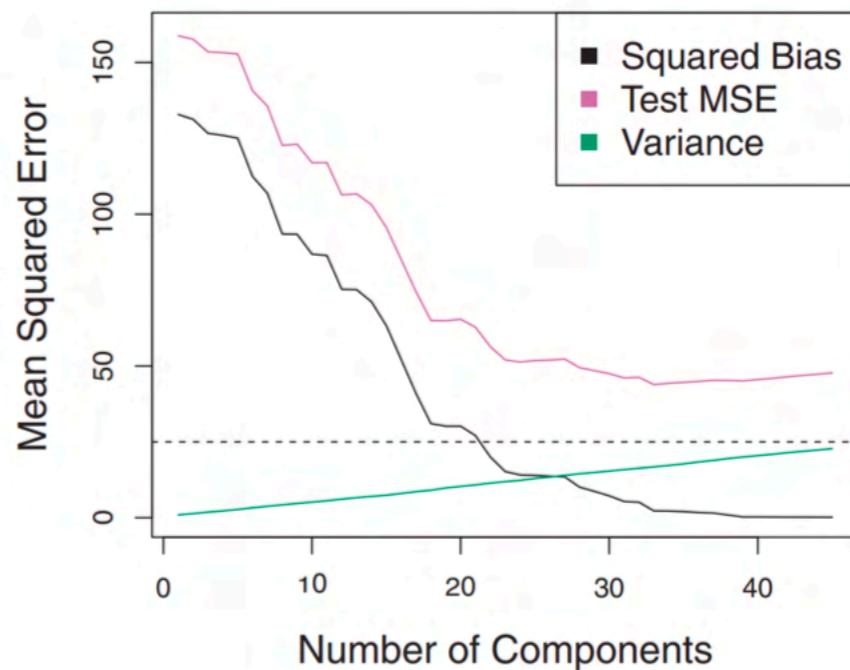
Choose Dimension

Proportion of the Variance Explained =
$$\frac{\sum_{m=1}^K S_{mm}}{\sum_{m=1}^p S_{mm}}$$

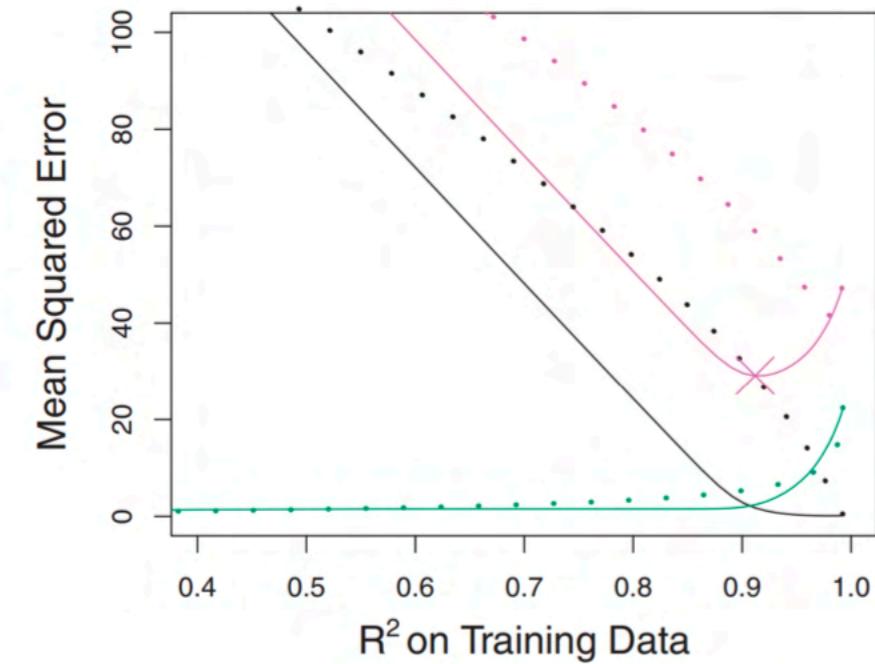


PCA Regression

Use PC scores as features for regression:



PCR



Ridge (dotted) & Lasso (solid)

Dimension Reduction: PCA Applications

Junwei Lu



HARVARD
T.H. CHAN

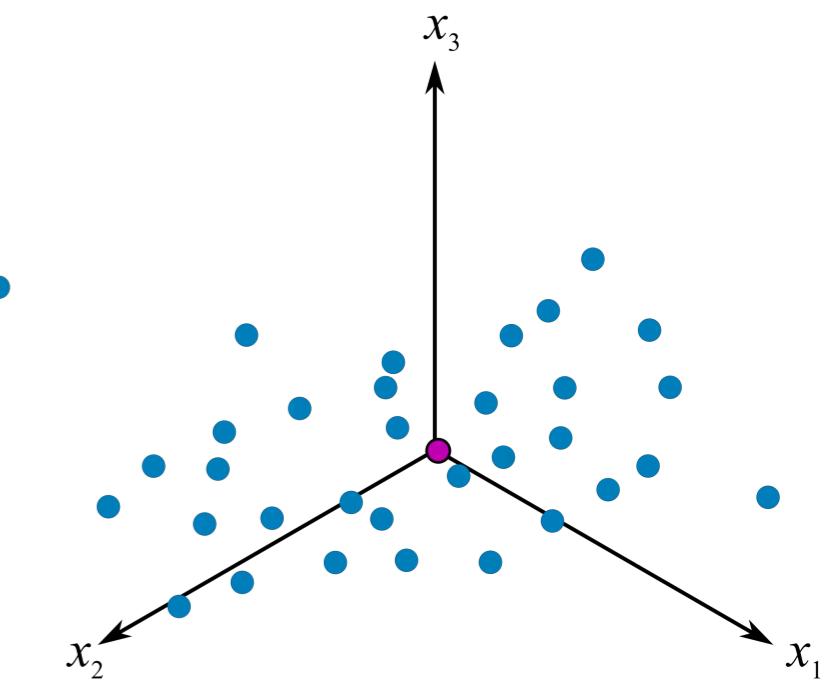
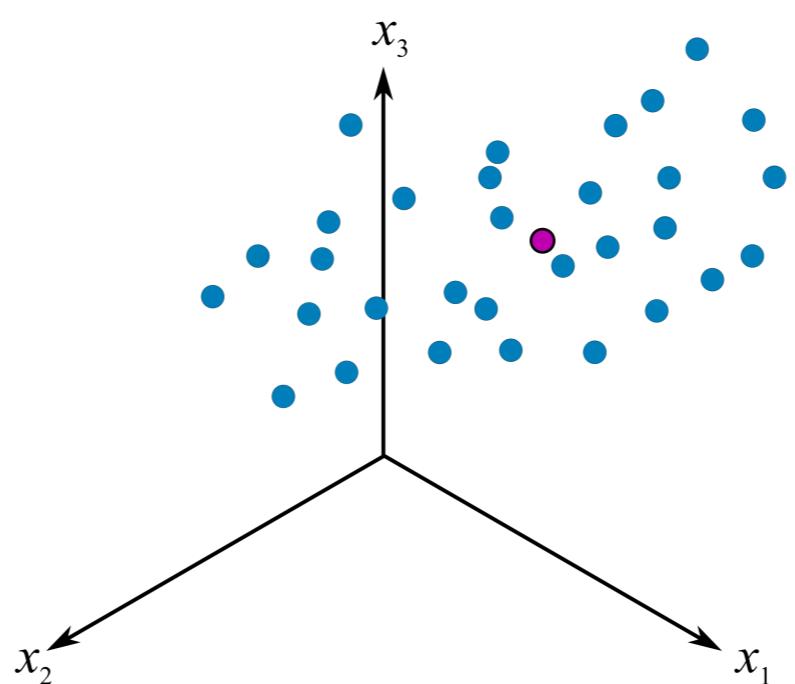
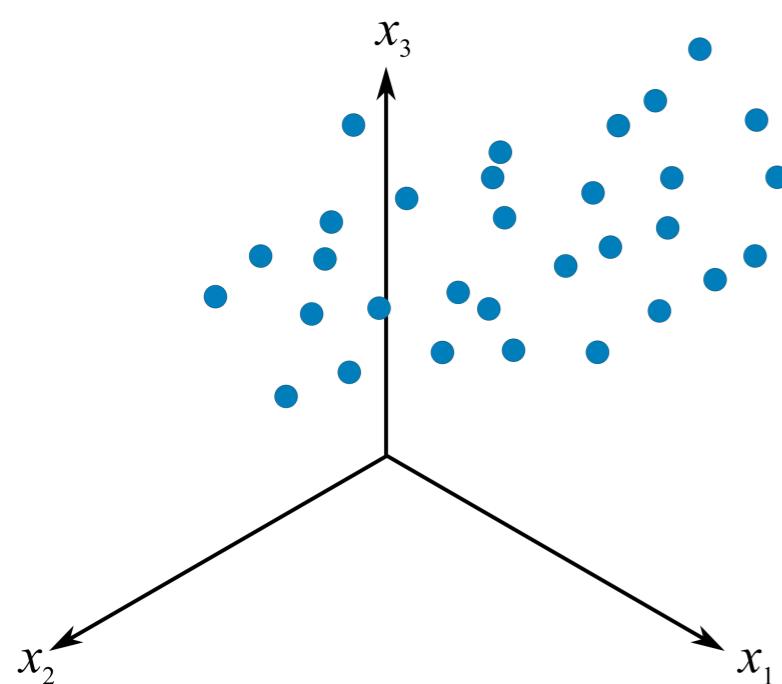
SCHOOL OF PUBLIC HEALTH
Department of Biostatistics



Principal Components Analysis (Review)

Step 1. Center the data so that $\frac{1}{n} \sum_{i=1}^n x_{ij} = 0$ for each j .

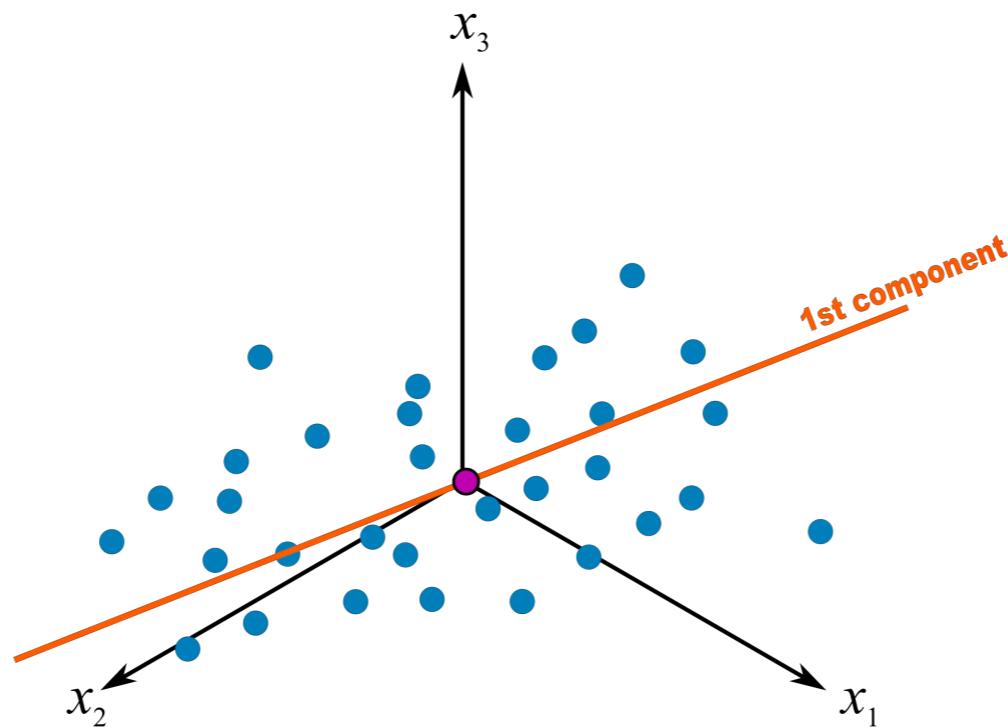
Step 2. Put the data into an $n \times p$ matrix: $X = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$.



Principal Components Analysis (Review)

PC direction:

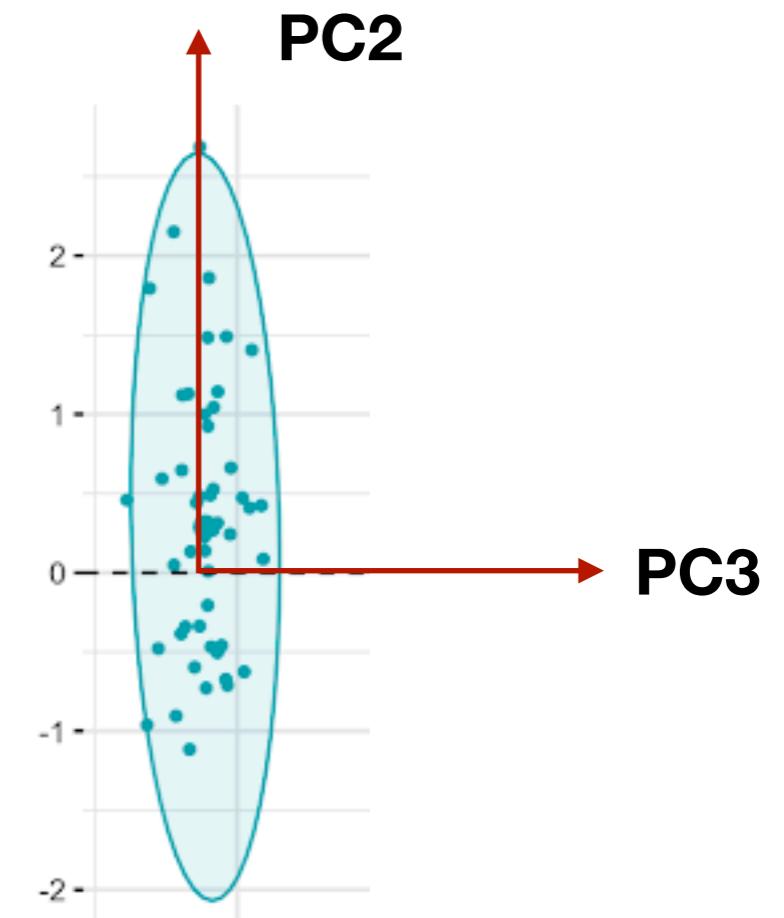
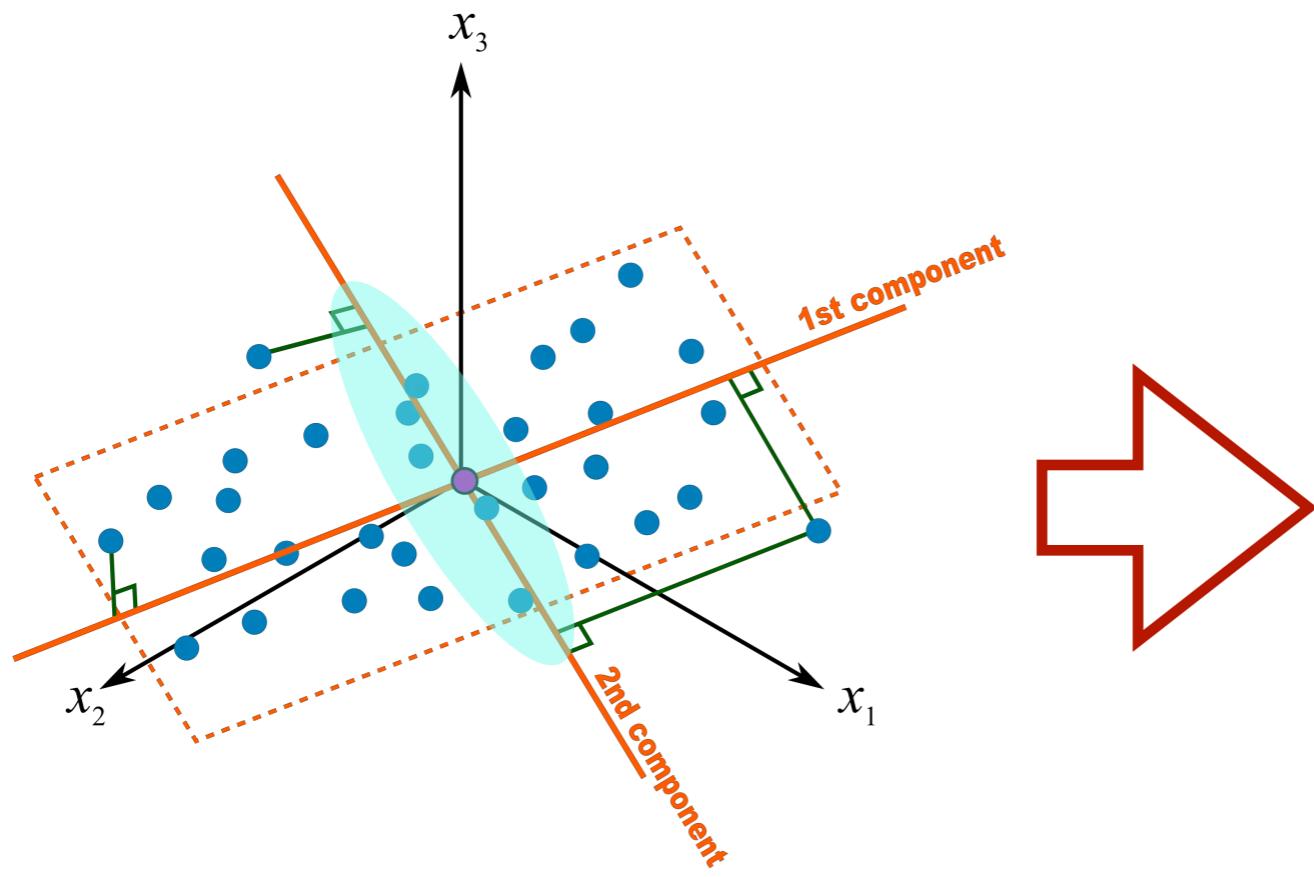
- *PC1 direction* is the direction along which the data has the largest variance.



Principal Components Analysis (Review)

PC direction:

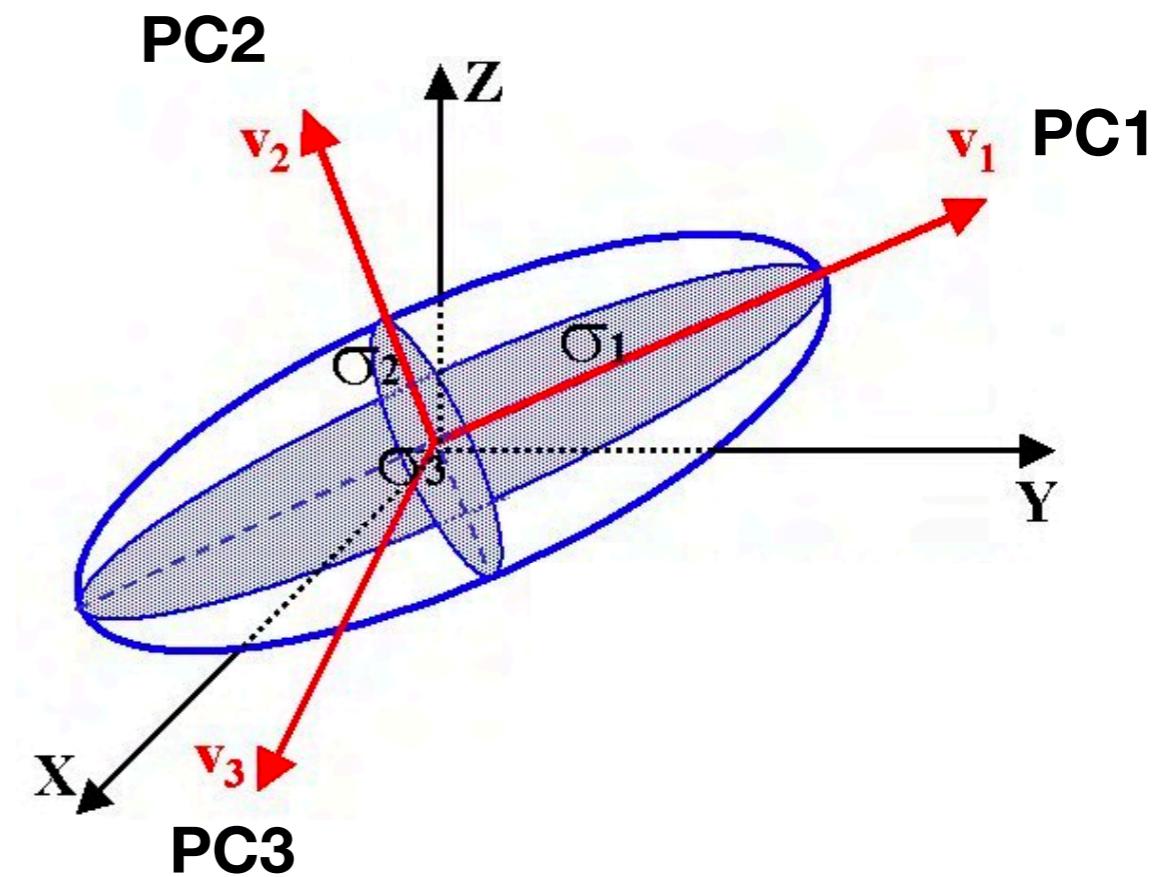
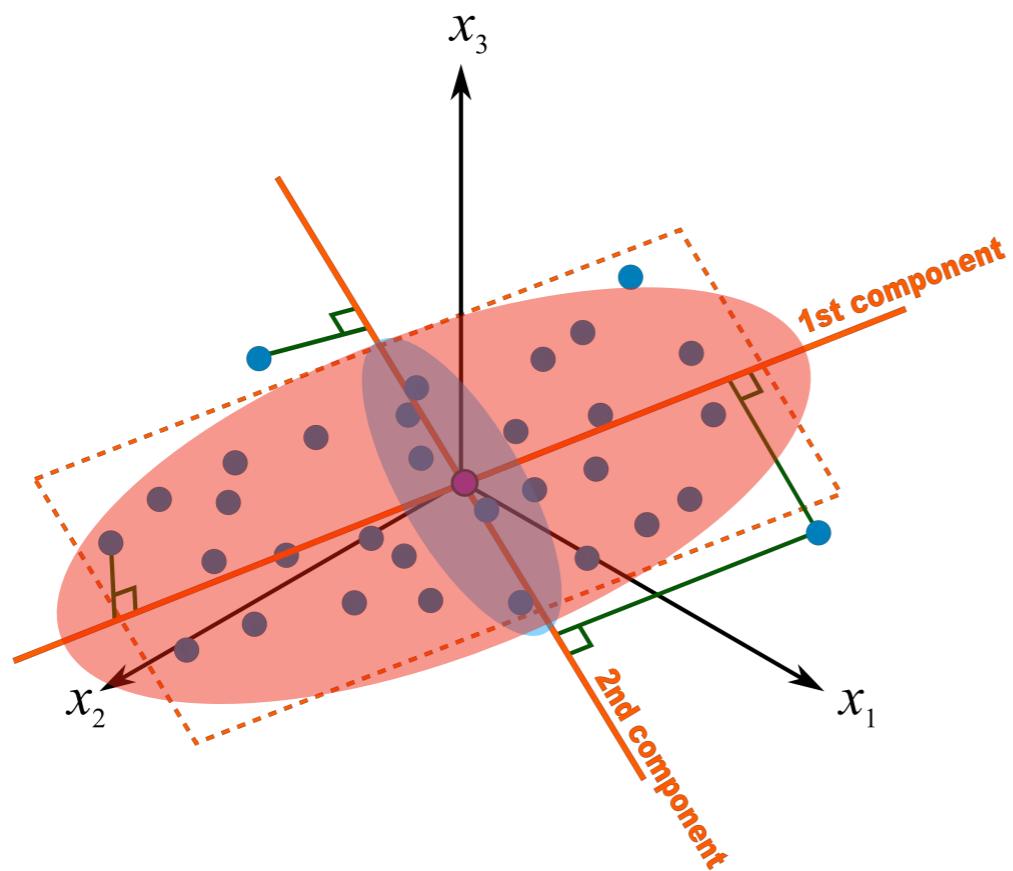
- *PC_m direction* is the direction along which the data has the largest variance, among all directions orthogonal to the first $m - 1$ PC directions.



Principal Components Analysis (Review)

PC direction:

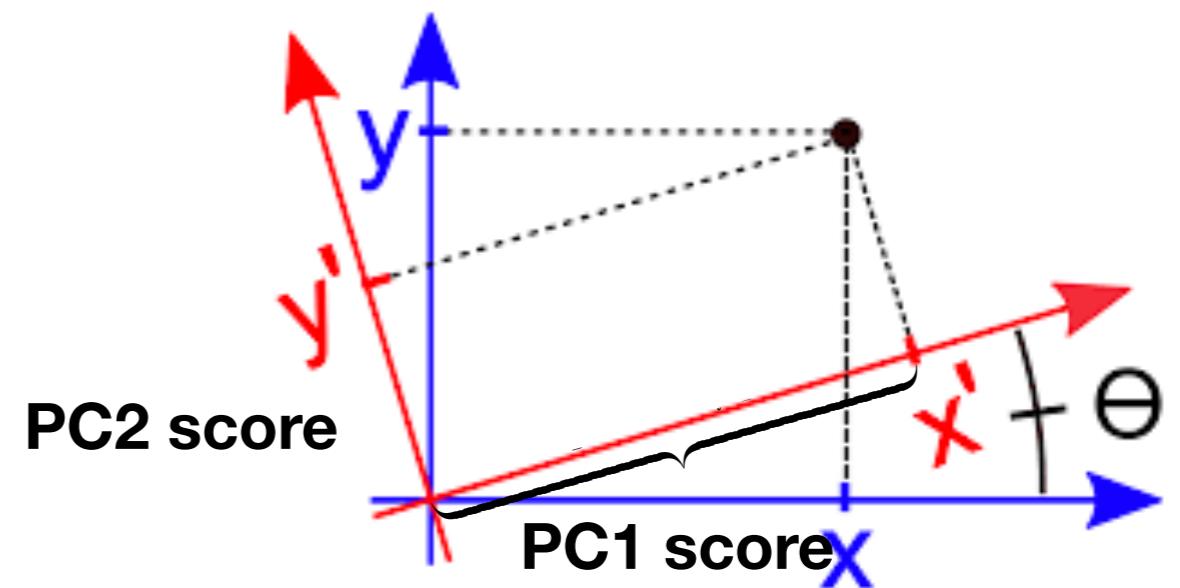
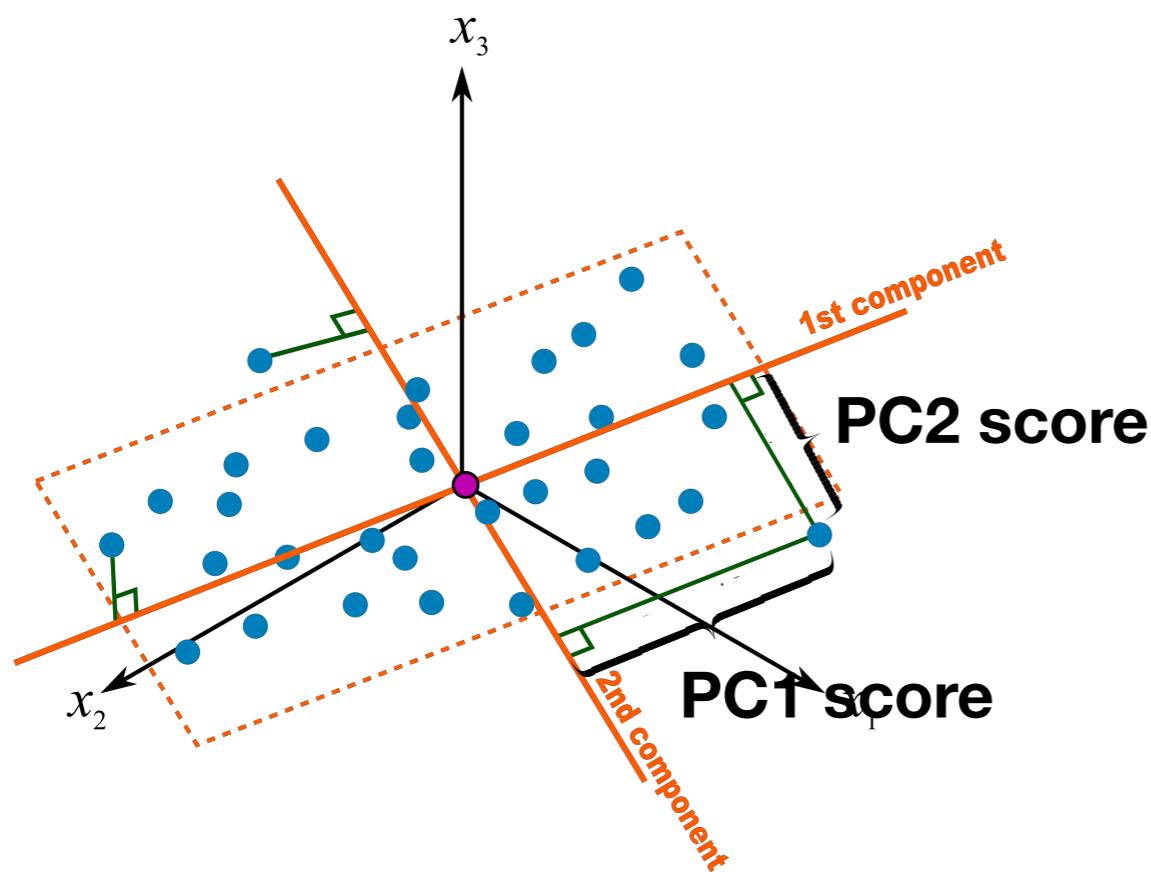
- *PC_m direction* is the direction along which the data has the largest variance, among all directions orthogonal to the first $m - 1$ PC directions.



Principal Components Analysis (Review)

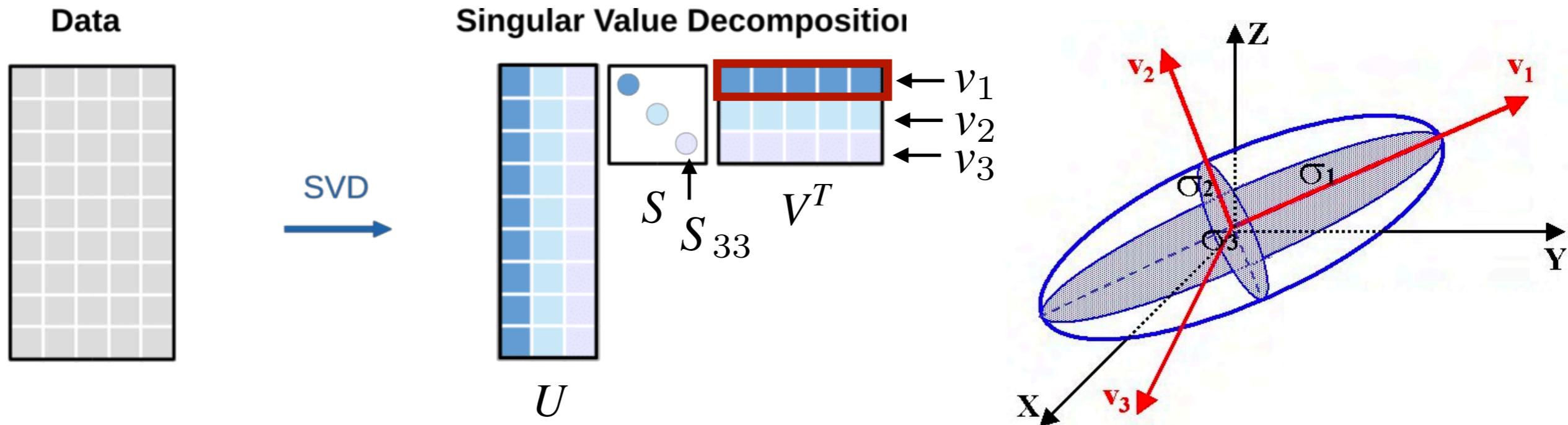
PC score:

- *PC_m score of a data point is its projection to the PC_m direction*



Principal Components Analysis (Review)

Step 3. Compute the SVD: $X = USV^T$.

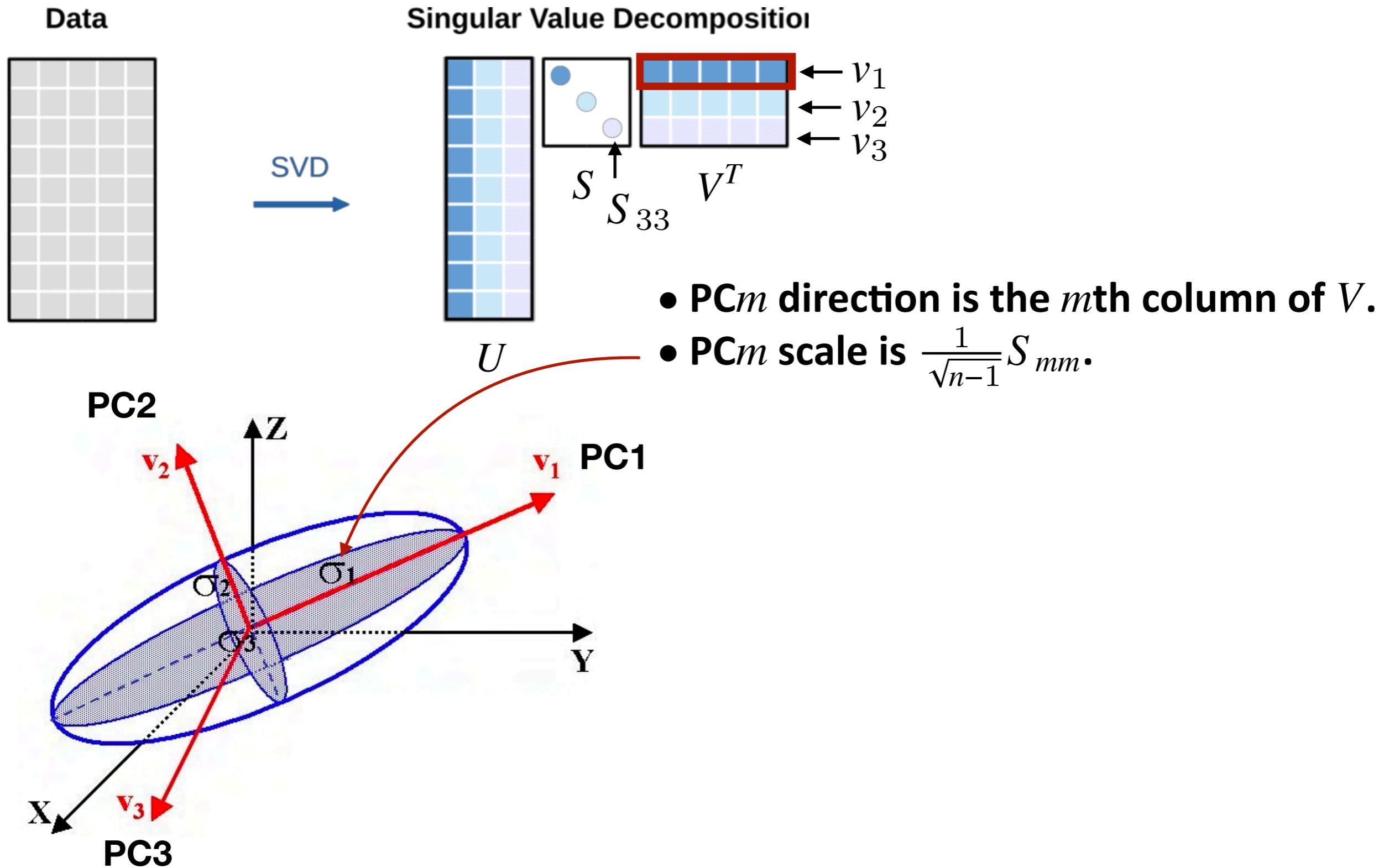


SVD of k -dimensions:

- $U \in \mathbb{R}^{n \times k}$ is **orthogonal**, i.e., $U^T U = I_k$
- $V \in \mathbb{R}^{p \times k}$ is **orthogonal**, i.e., $V^T V = I_k$
- $S \in \mathbb{R}^{k \times k}$ is zero everywhere except $S_{11} \geq S_{22} \geq \dots \geq 0$.

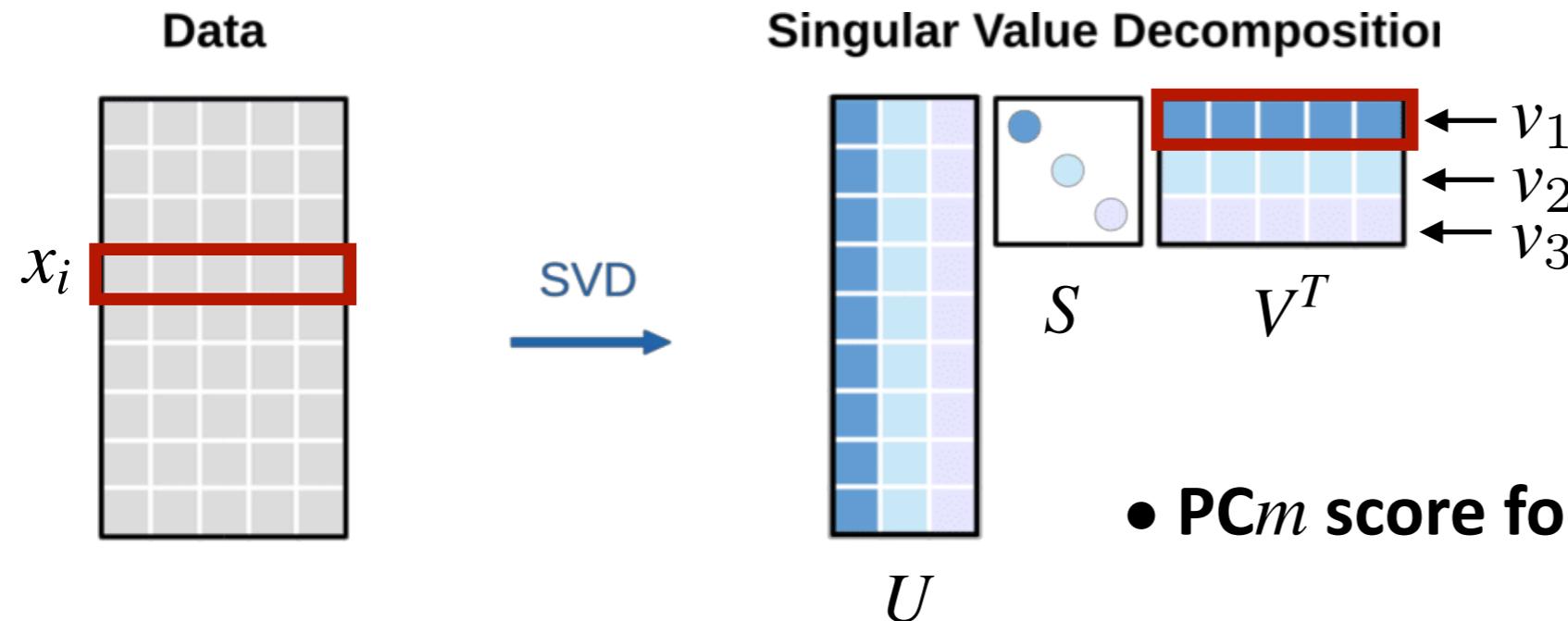
Principal Components Analysis (Review)

Step 3. Compute the SVD: $X = USV^T$.

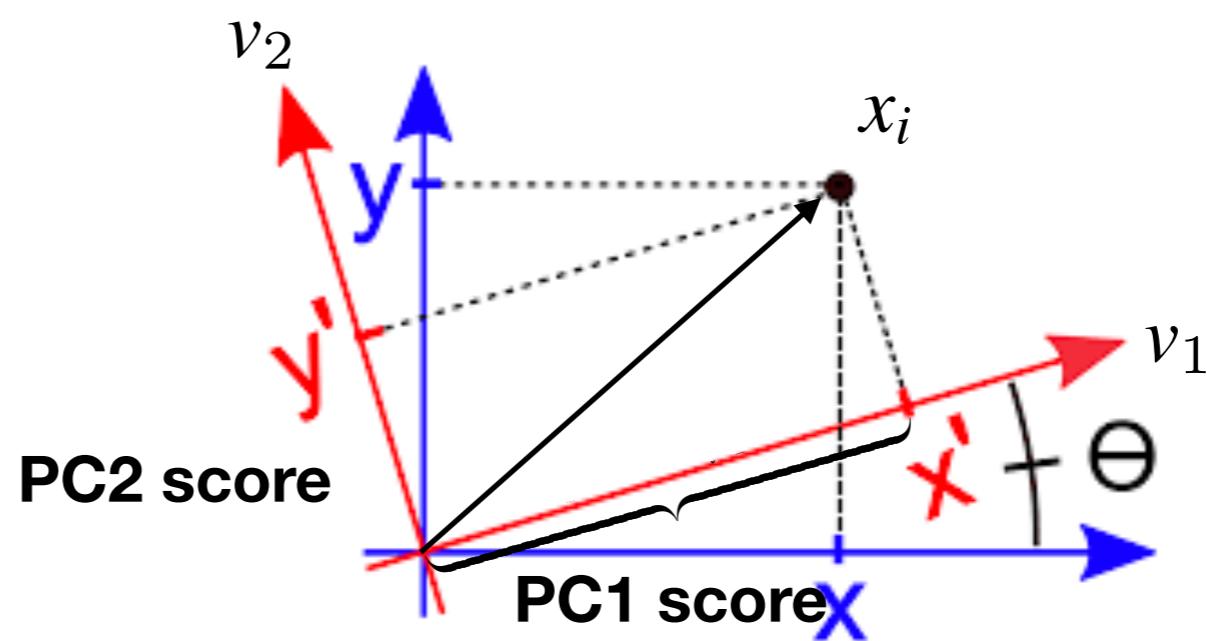


Principal Components Analysis (Review)

Step 3. Compute the SVD: $X = USV^T$.

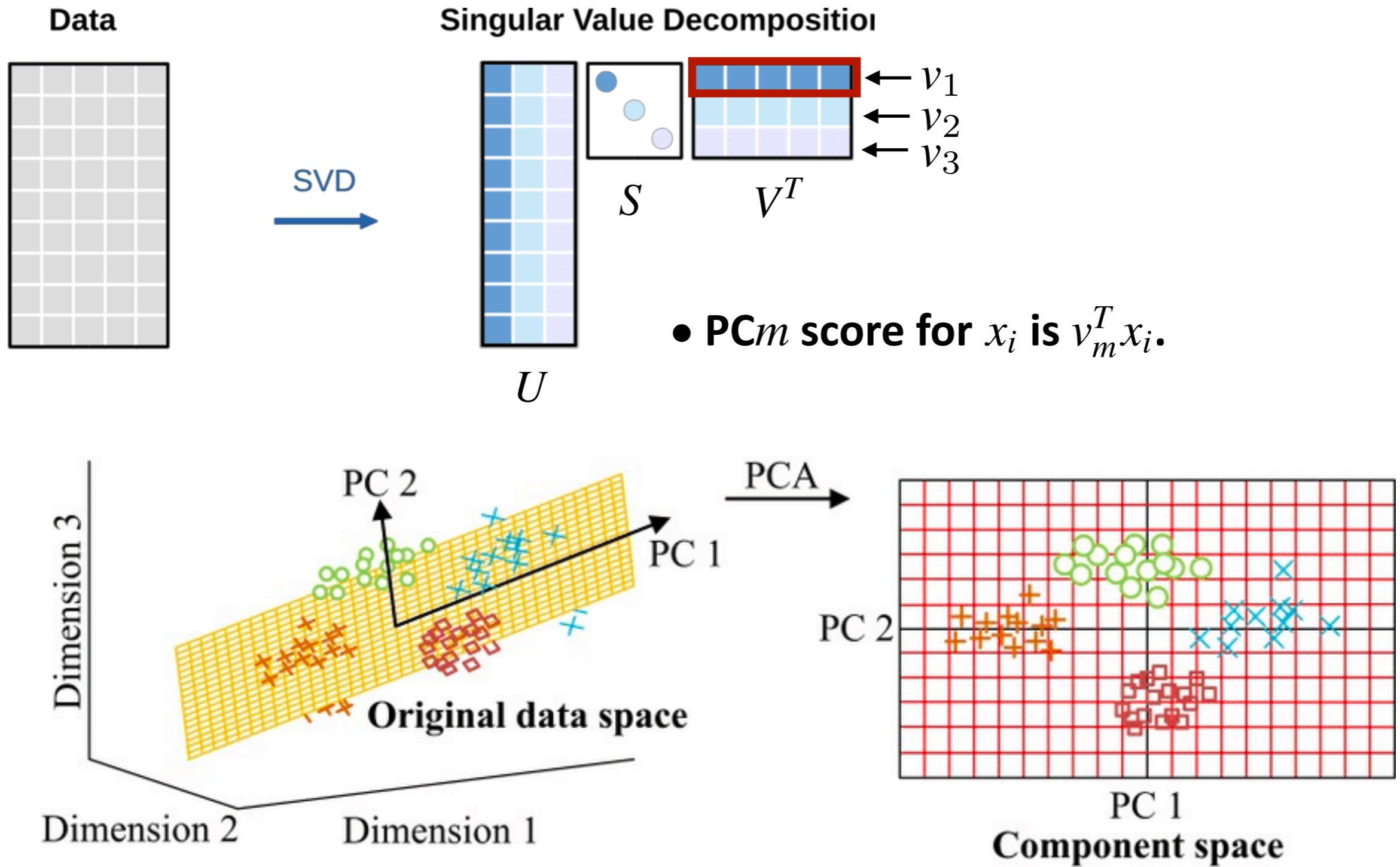


- PC m score for x_i is $v_m^T x_i$.

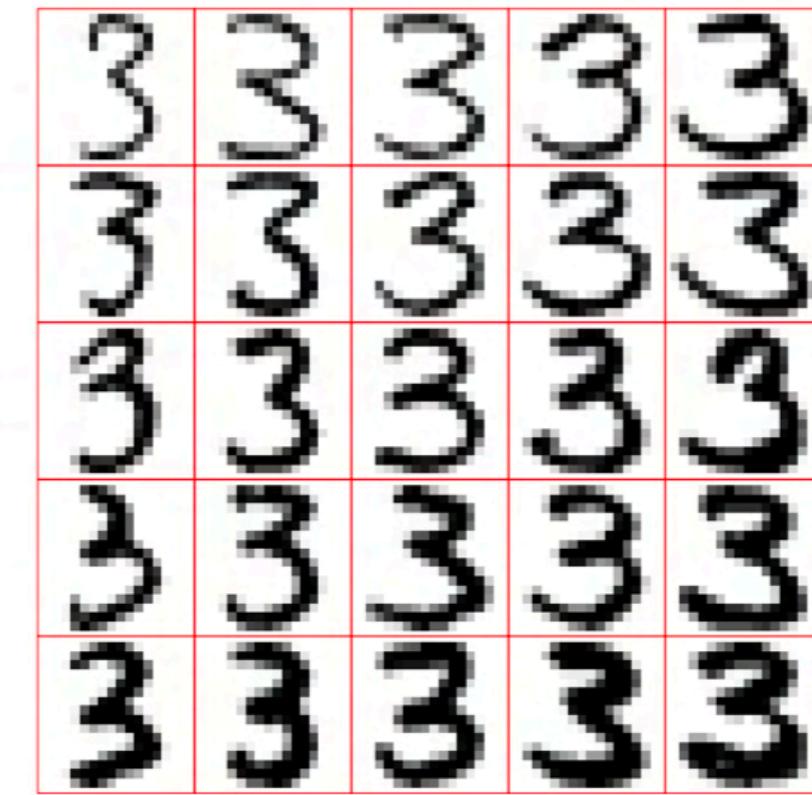
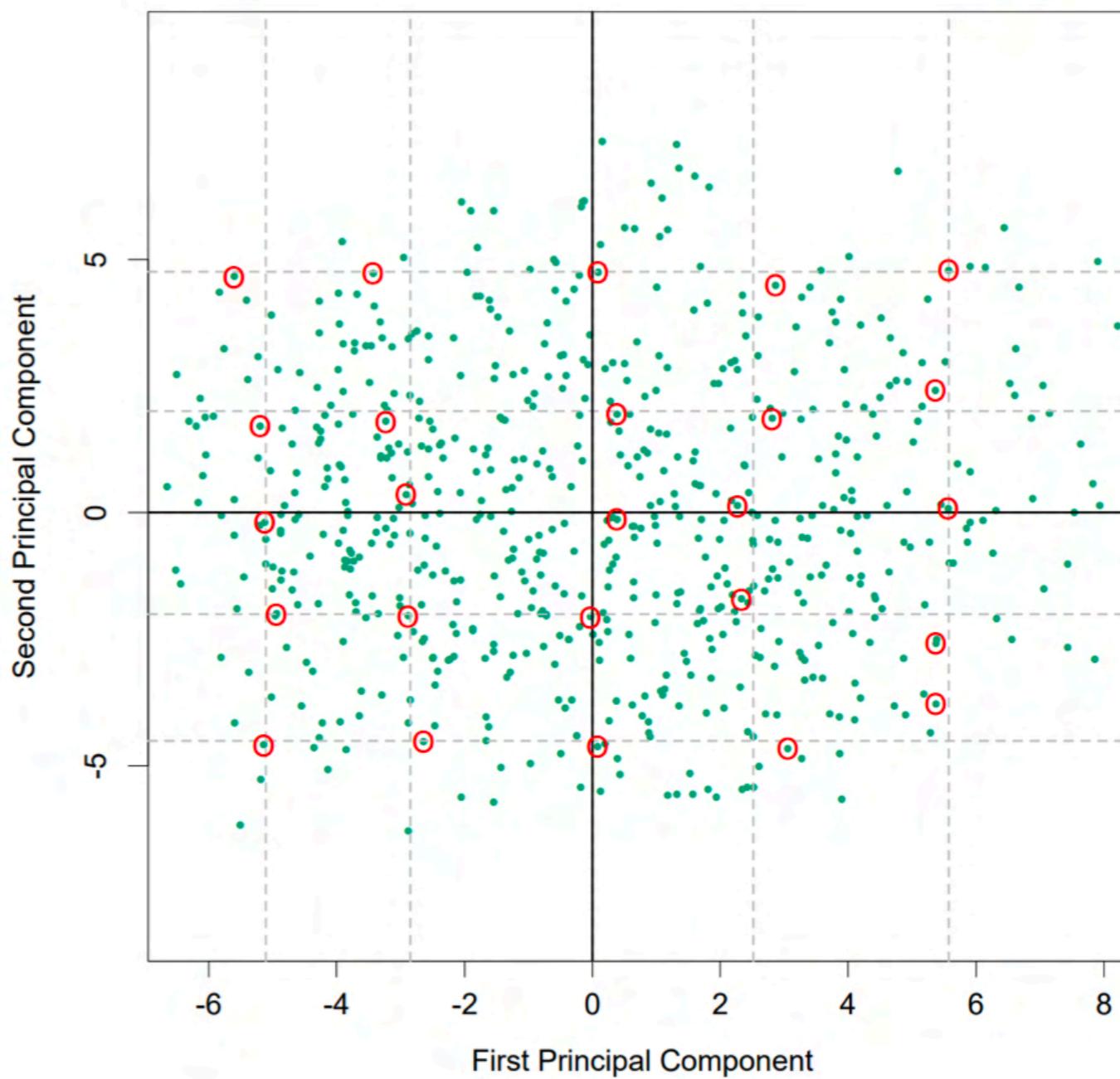


Principal Components Analysis (Review)

Step 3. Compute the SVD: $X = USV^T$.



PCA Example: Hand Writing Digits



PCA can be used to make a low-dimensional approximation to each image.

Application to PCA

- Clustering
- Batch effect
- Topic Model
- Social network
- Recommendation system

PCA for Biomarker Selection

PCA in subtype diagnosis for genomic data

- Identify genes with high variance
- Perform PCA on them
- Plot using PC1 to PC3

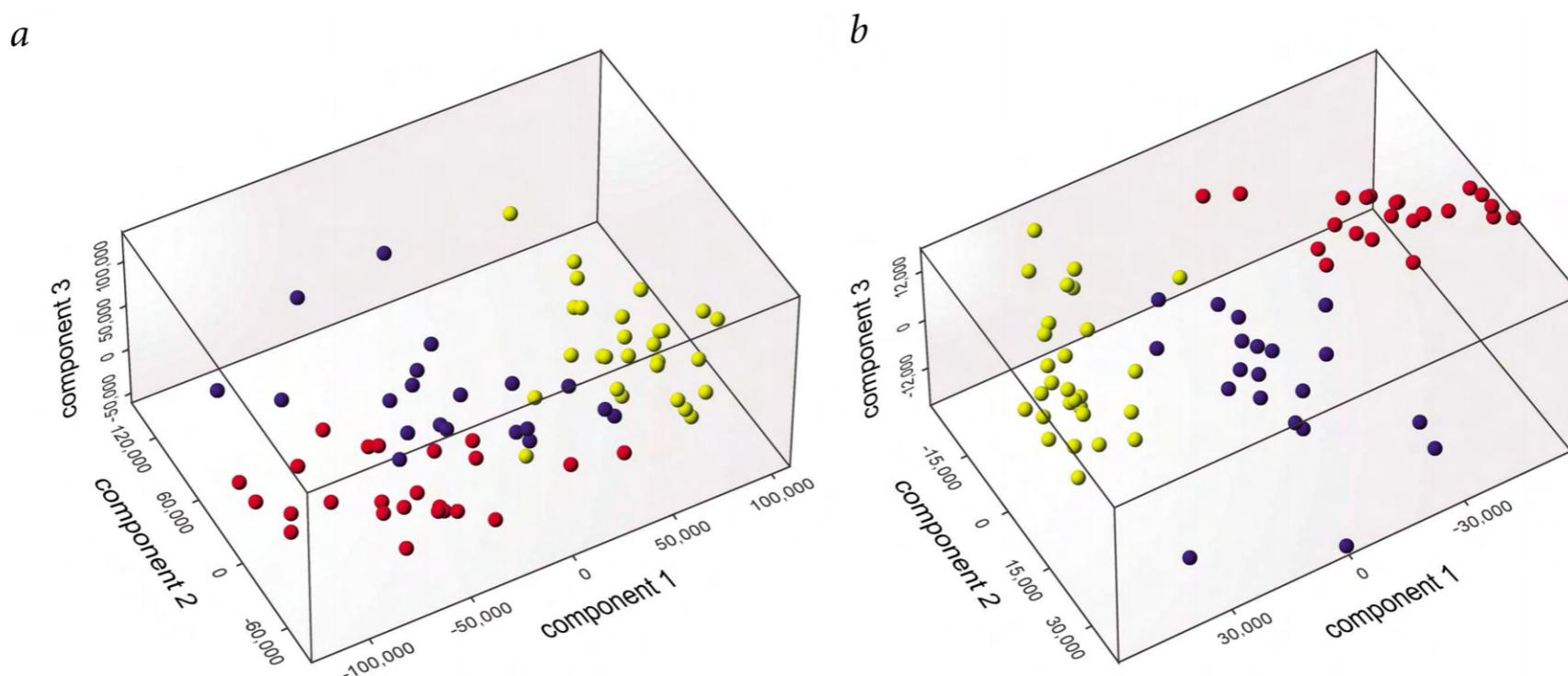
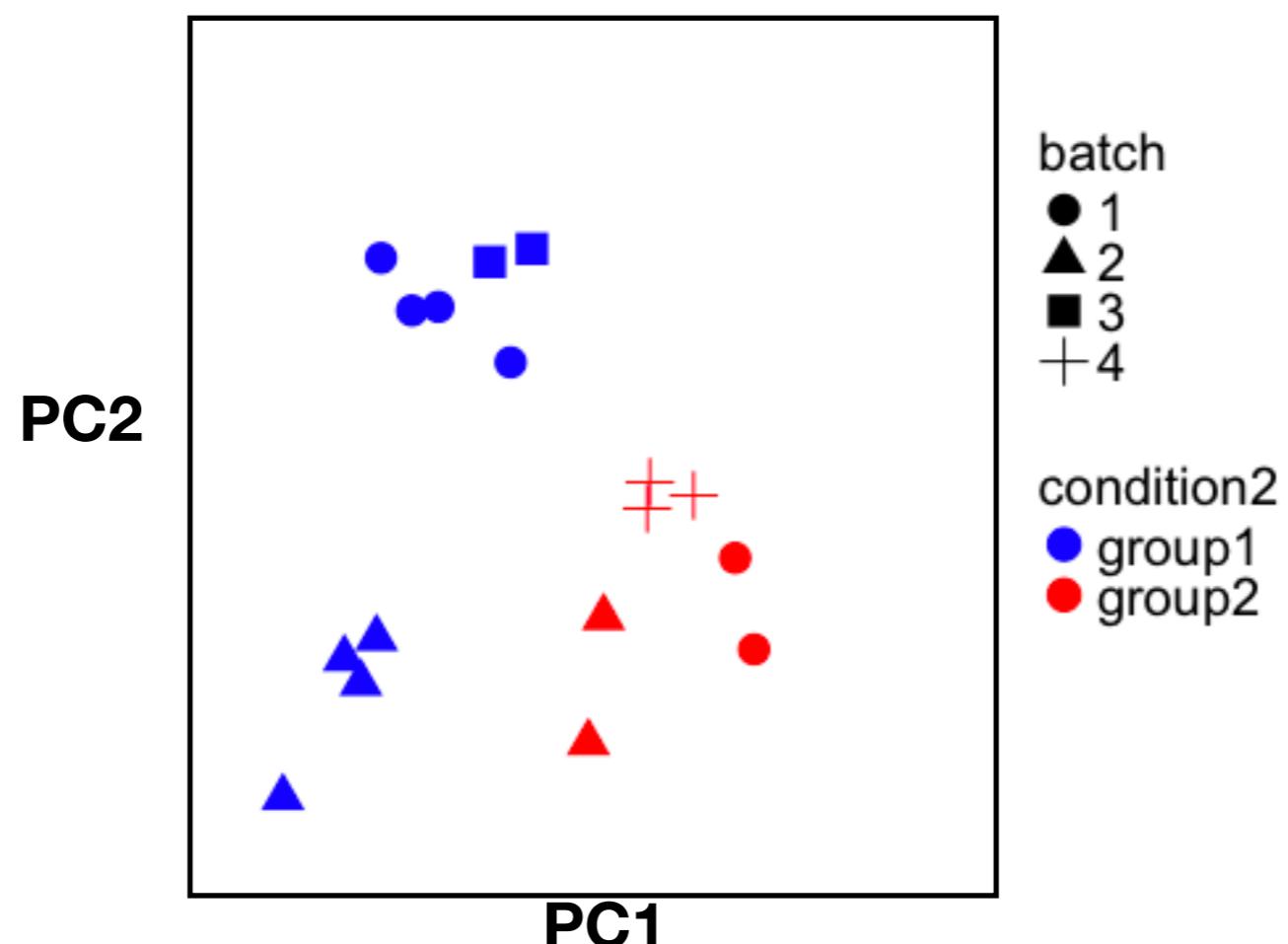


Fig. 4 Comparison of gene expression between ALL, MLL and AML. **a**, Principal component analysis (PCA) plot of ALL (red), MLL (blue) and AML (yellow) carried out using 8,700 genes that passed filtering. **b**, PCA plot comparing ALL (red), MLL (blue) and AML (yellow) using the 500 genes that best distinguished ALL from AML. Three-dimensional virtual reality modeling language (VRML) plots can be viewed at our web site (<http://research.dfci.harvard.edu/korsmeyer/MLL.htm>).

PCA to Remove Batch Effect

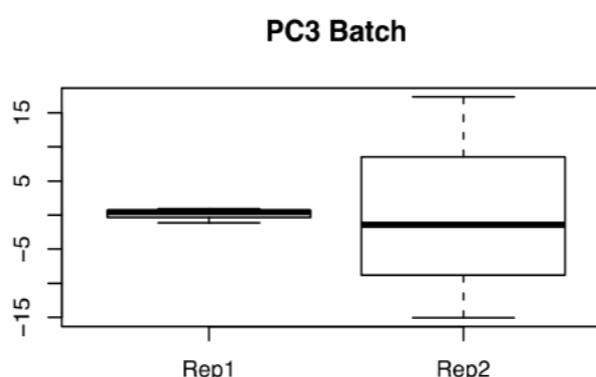
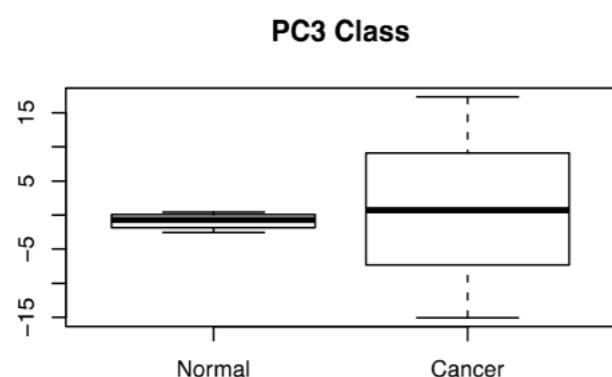
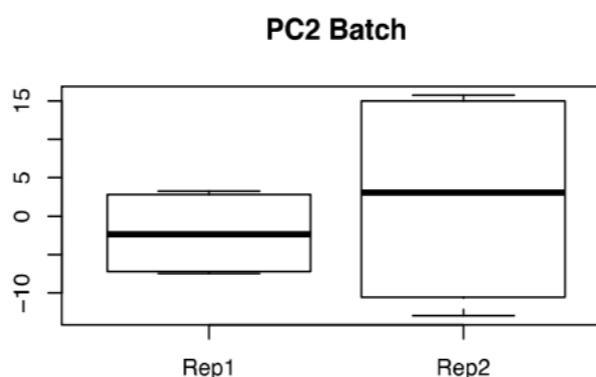
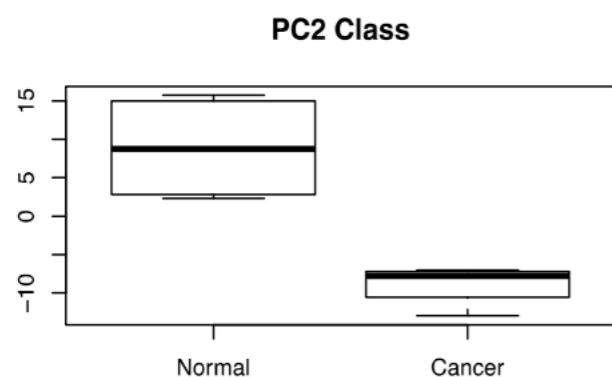
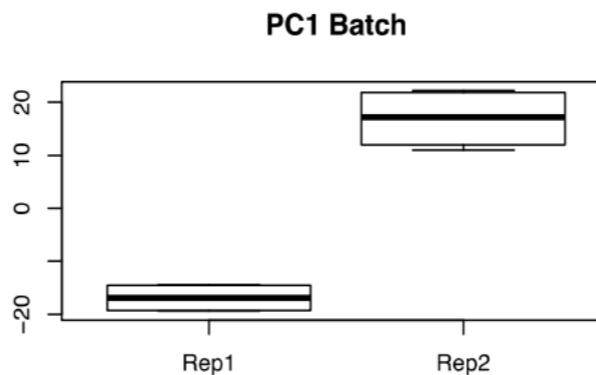
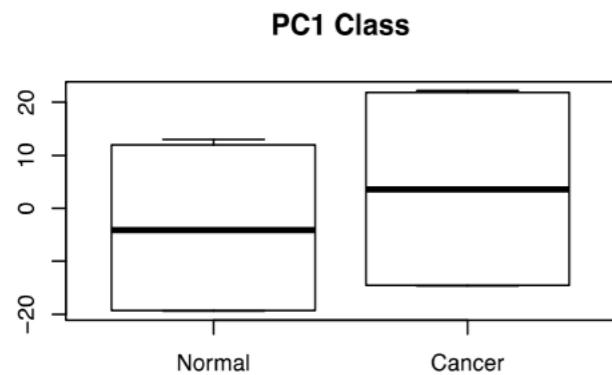
Batch effects are unwanted sources of variation caused by different processing date, handling personnel, reagent lots, equipment/machines,

Samples from diff batches are grouped together, regardless of subtypes and treatment response



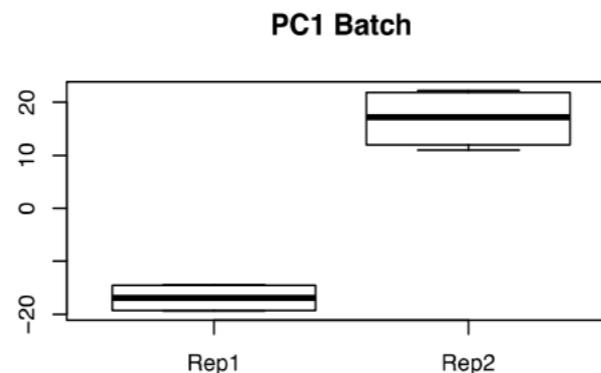
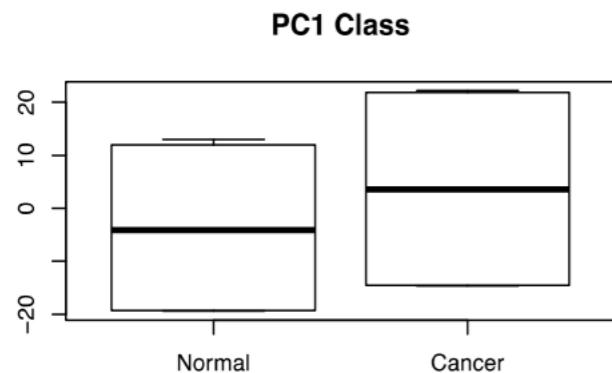
PCA to Remove Batch Effect

It is easier to see which PC is enriched in batch effects by showing, side by side, the distribution of values of each PC stratified by class or batch

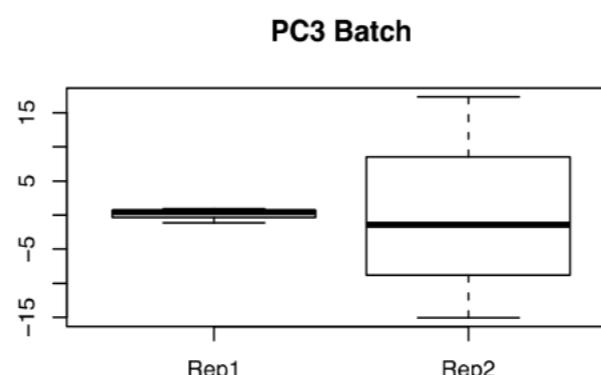
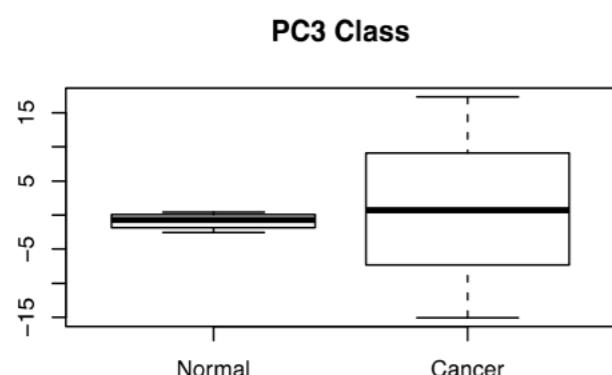
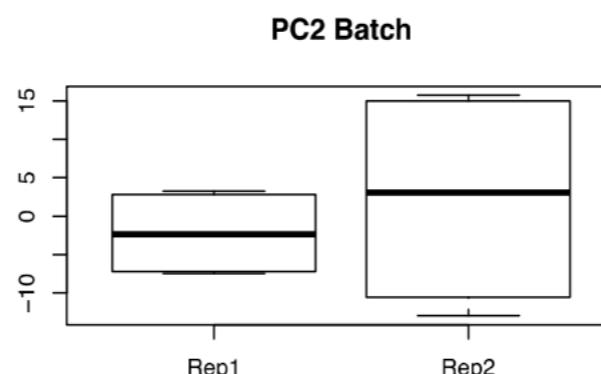
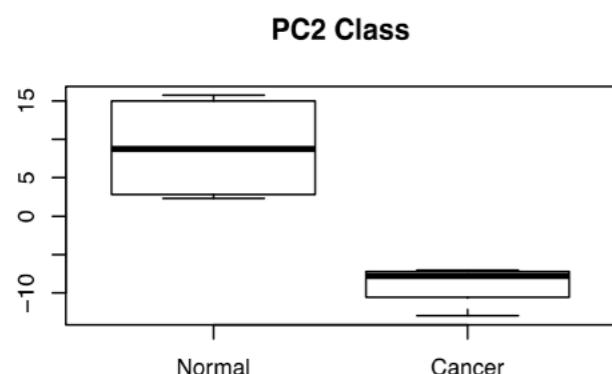


PCA to Remove Batch Effect

It is easier to see which PC is enriched in batch effects by showing, side by side, the distribution of values of each PC stratified by class or batch



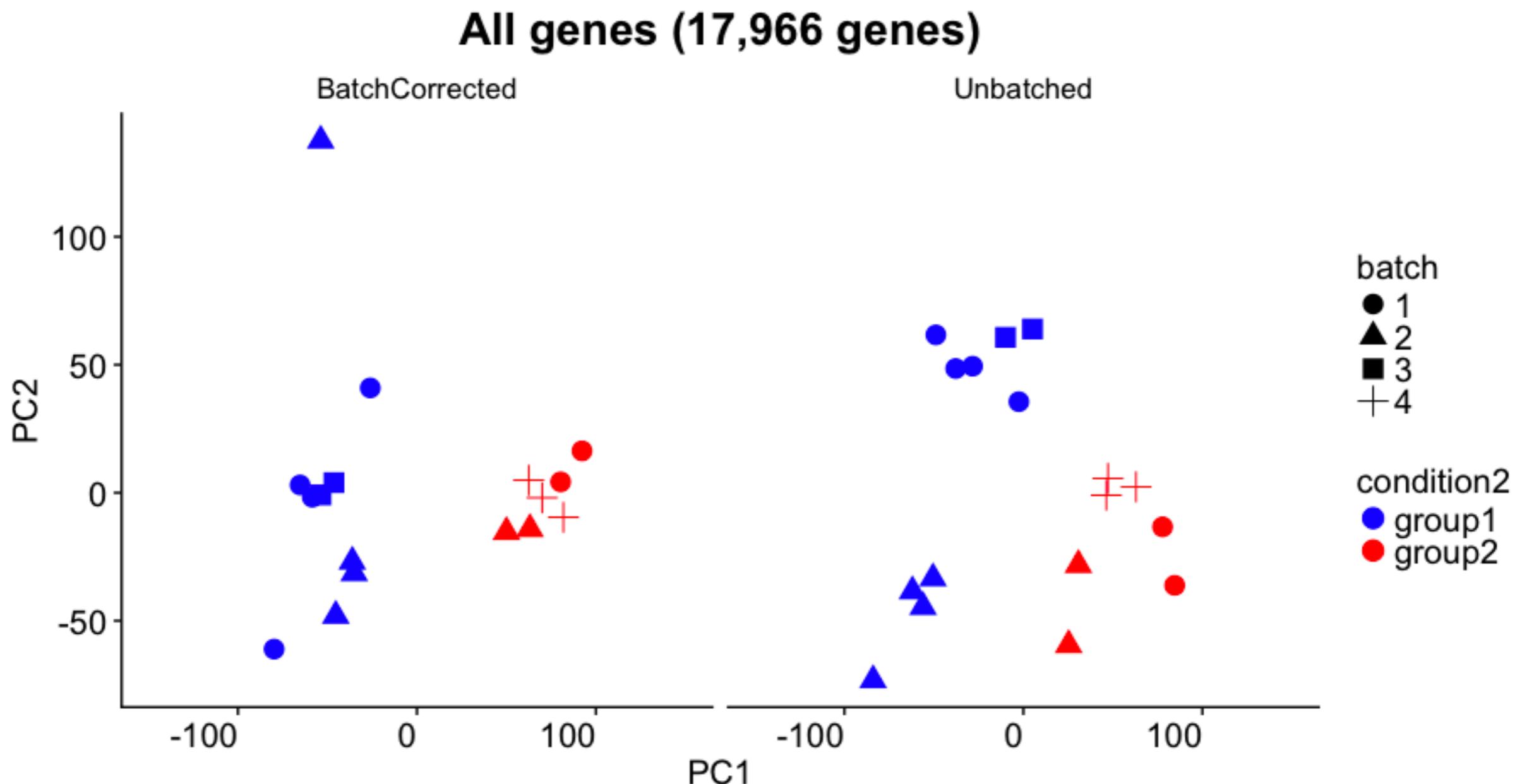
PC1 is batch effect!



PCA to Remove Batch Effect

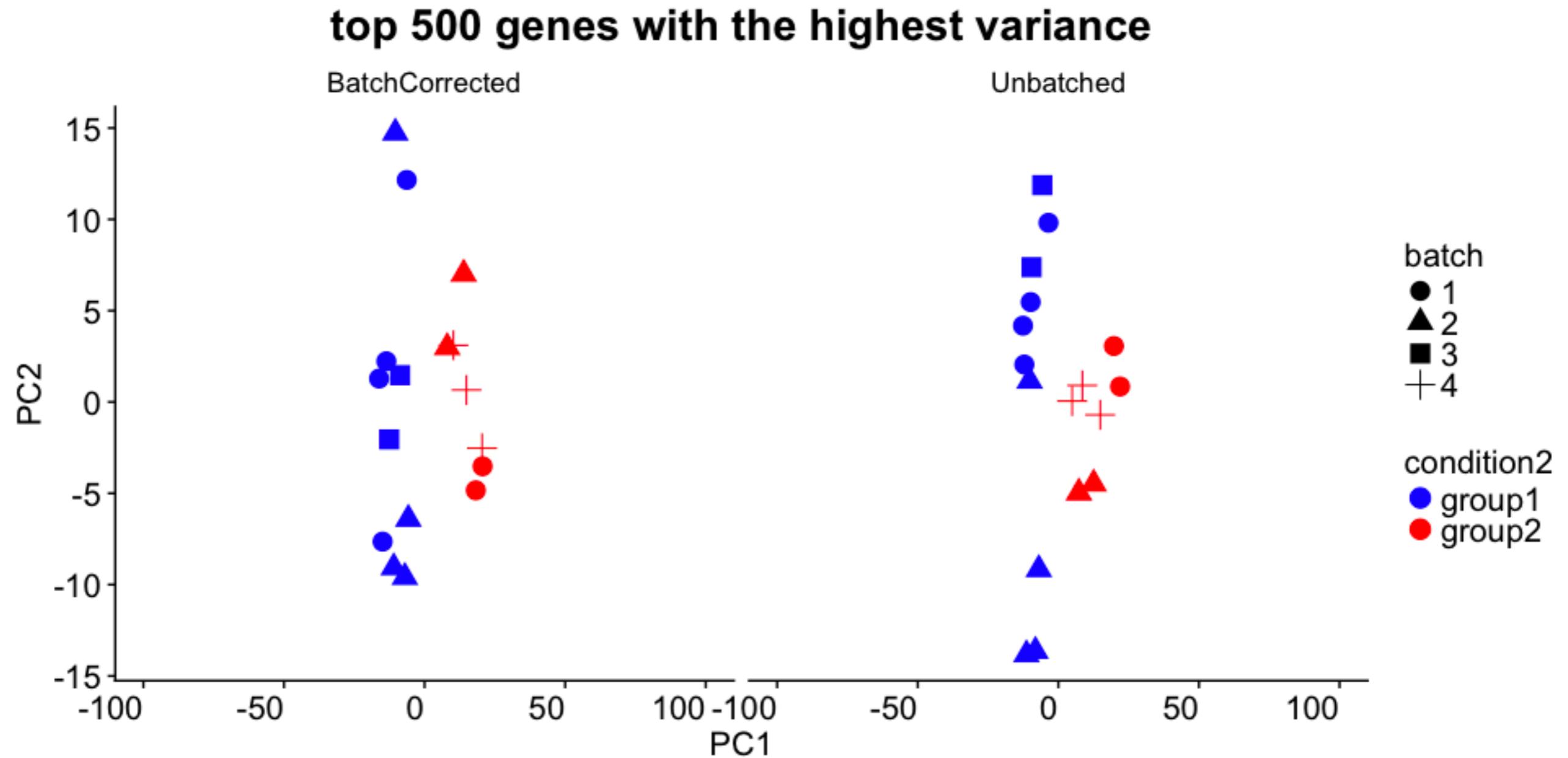
We aim to remove batch effect via normalizing PC's related to batch:

$$\text{PC1 score new} = (\text{PC1 score} - \text{mean of PC1 scores})/\text{Std of PC1 scores}$$



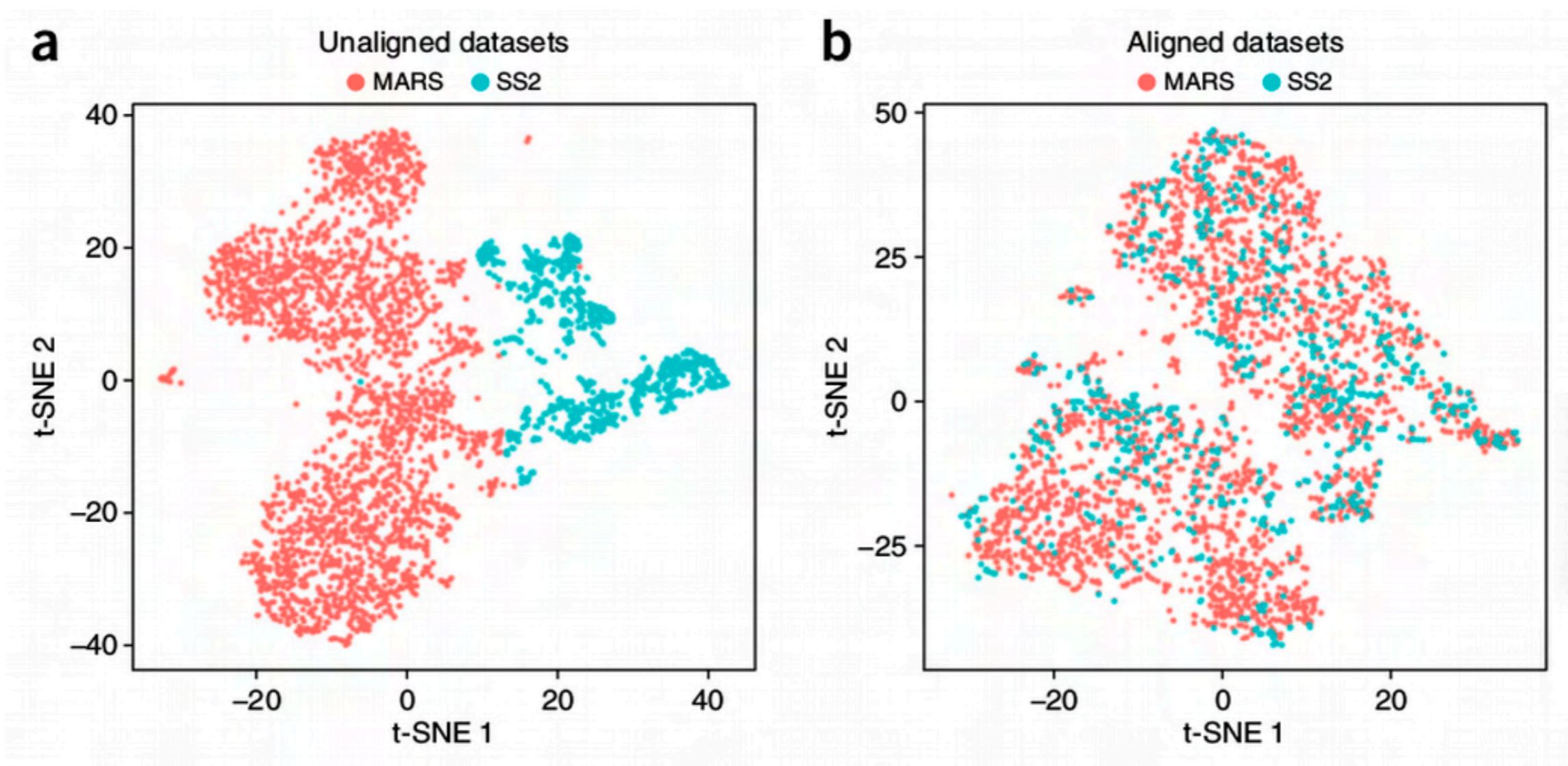
PCA to Remove Batch Effect

The performance will be better if fewer genes are involved



PCA to Remove Batch Effect

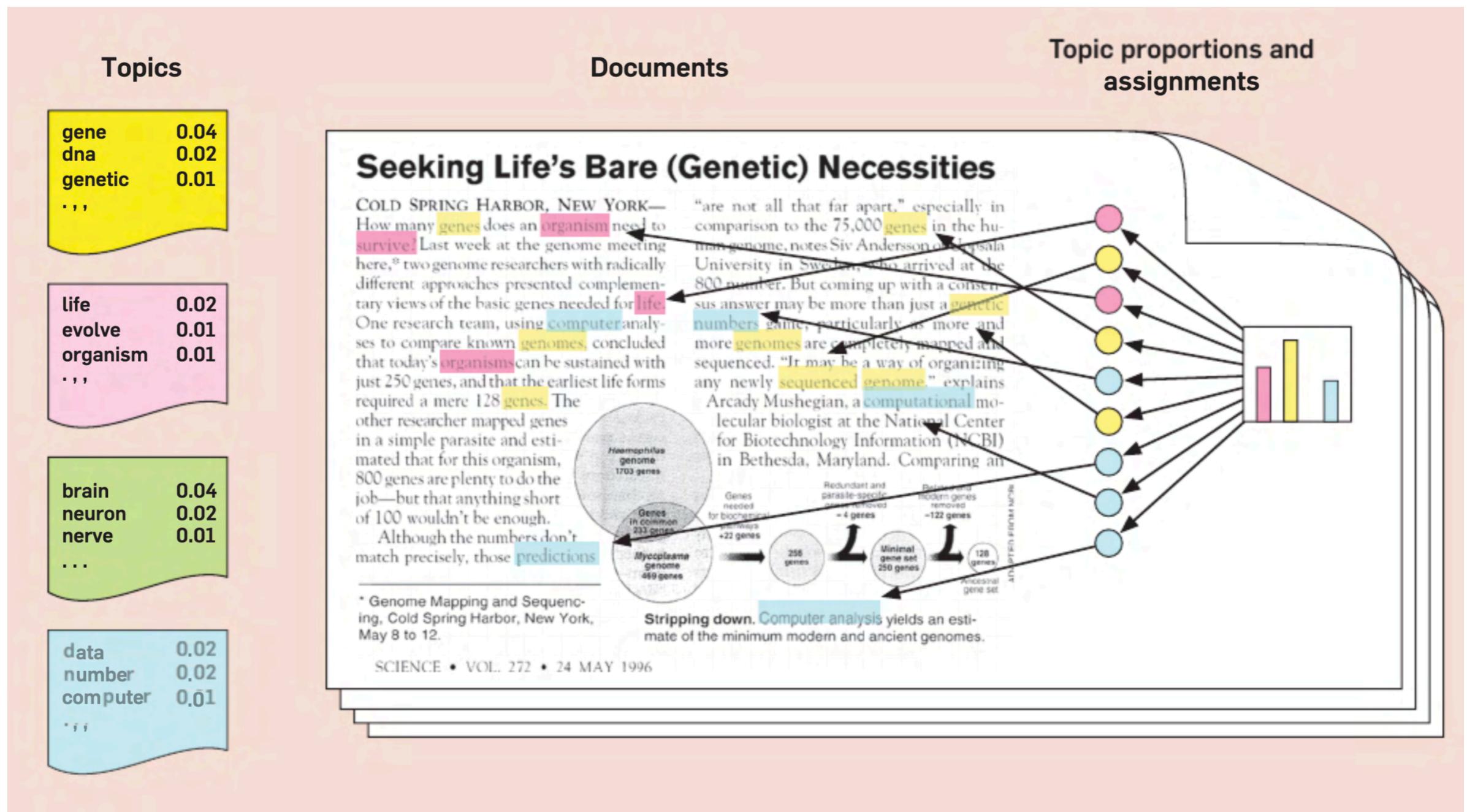
From Butler et al., Nature Biotechnology 36, p. 411–420 (2018)



Topic Model

We have n documents and m words.

Goal: Find the topic of each documents

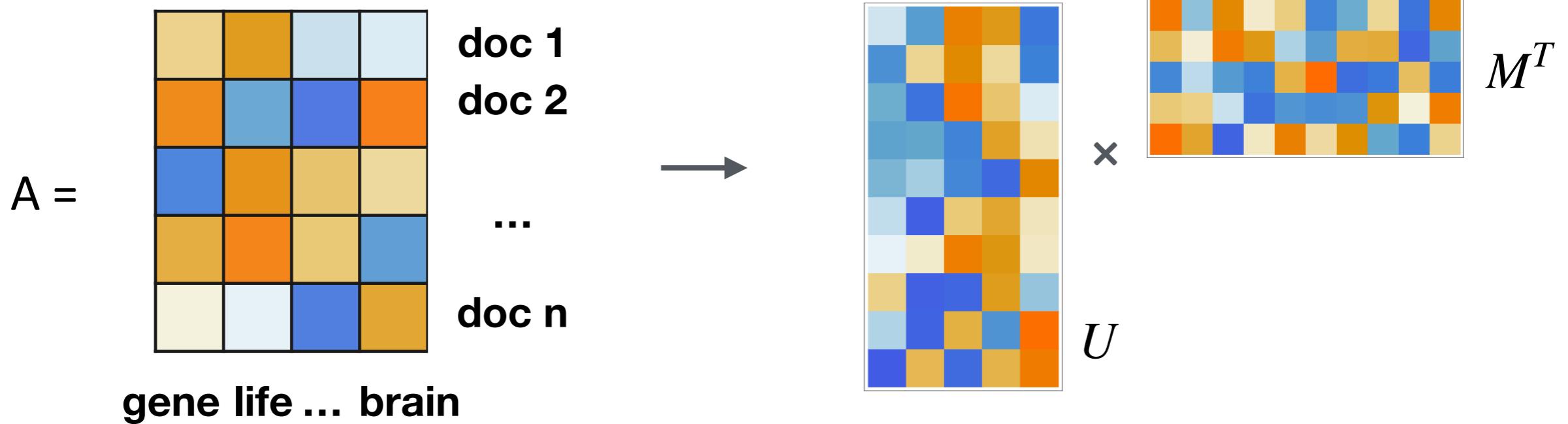


Topic Model

We have n documents and m words.

Goal: Find the topic of each documents

$$A_{ij} = \text{Freq of Word } j \text{ in Doc } i$$

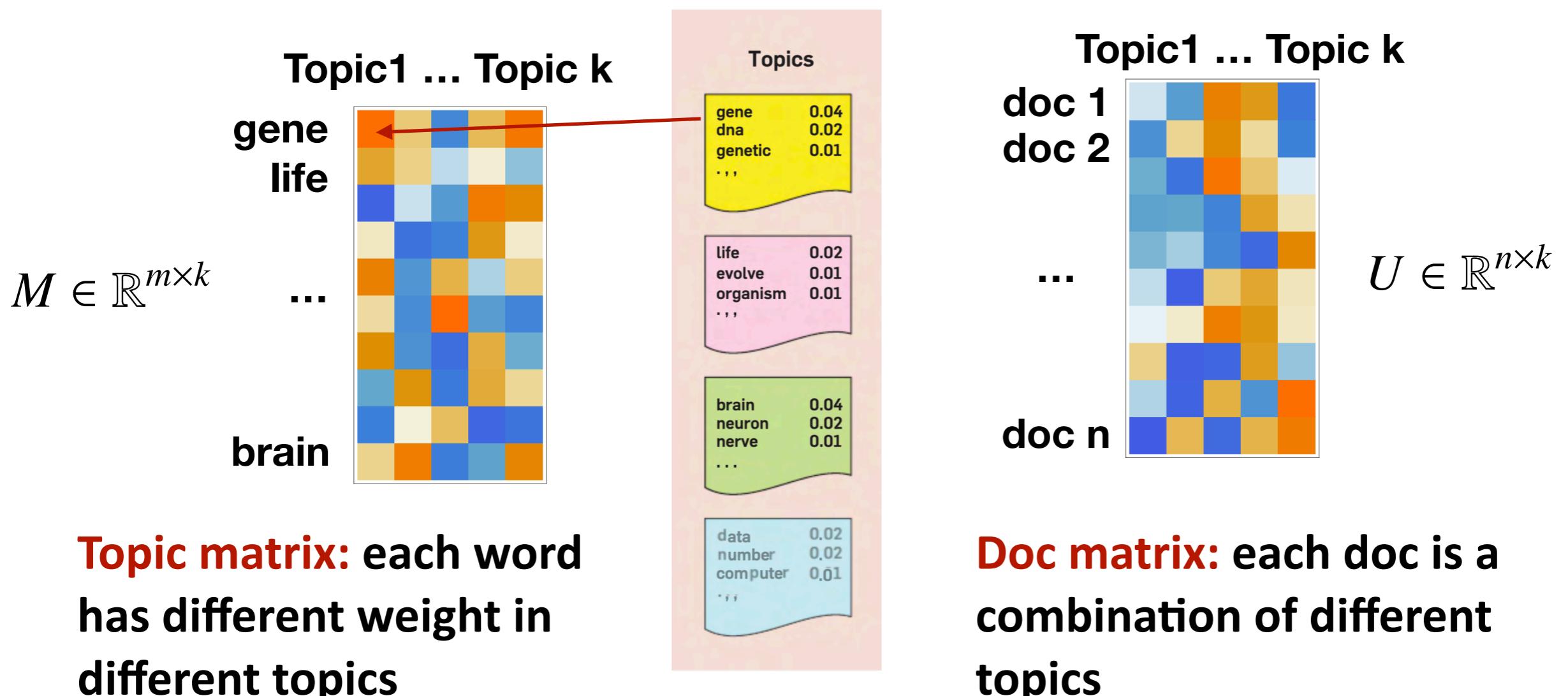


Topic Model

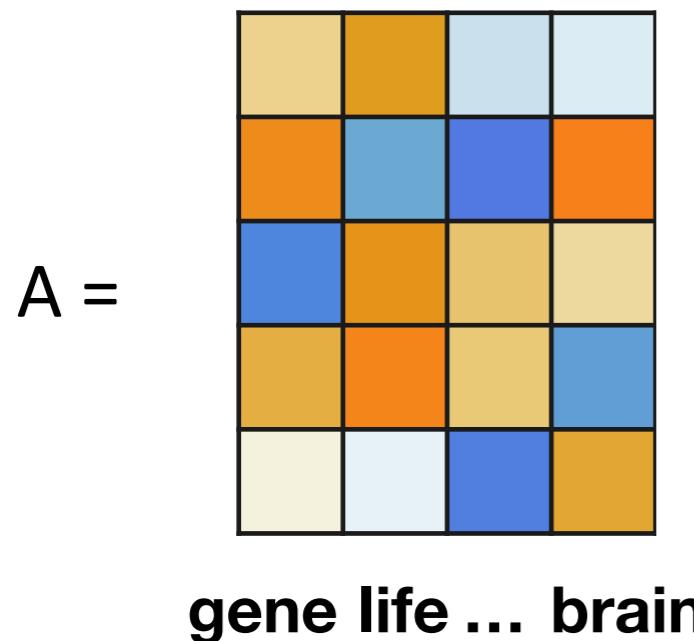
We have n documents and m words.

Goal: Find the topic of each documents

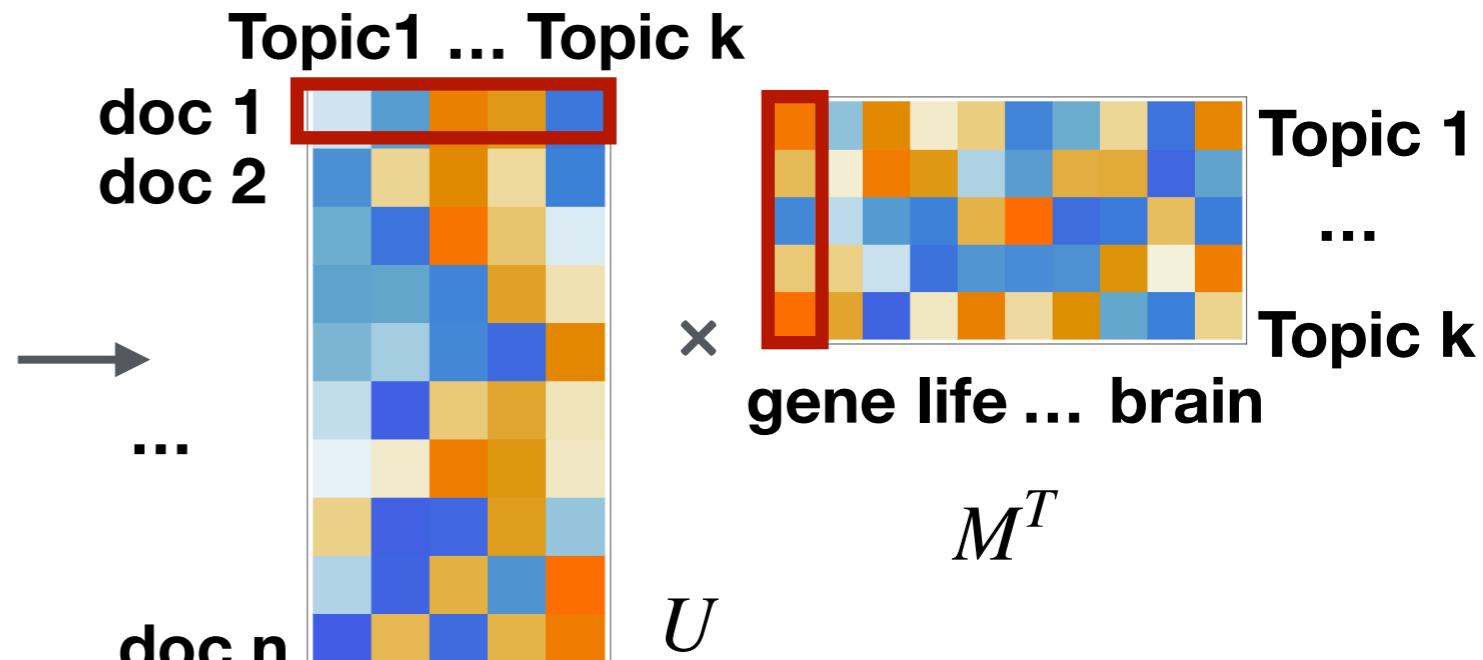
$$\text{SVD } A = USV^T, \text{ and } M = VS$$



Topic Model



doc 1
doc 2
...
doc n



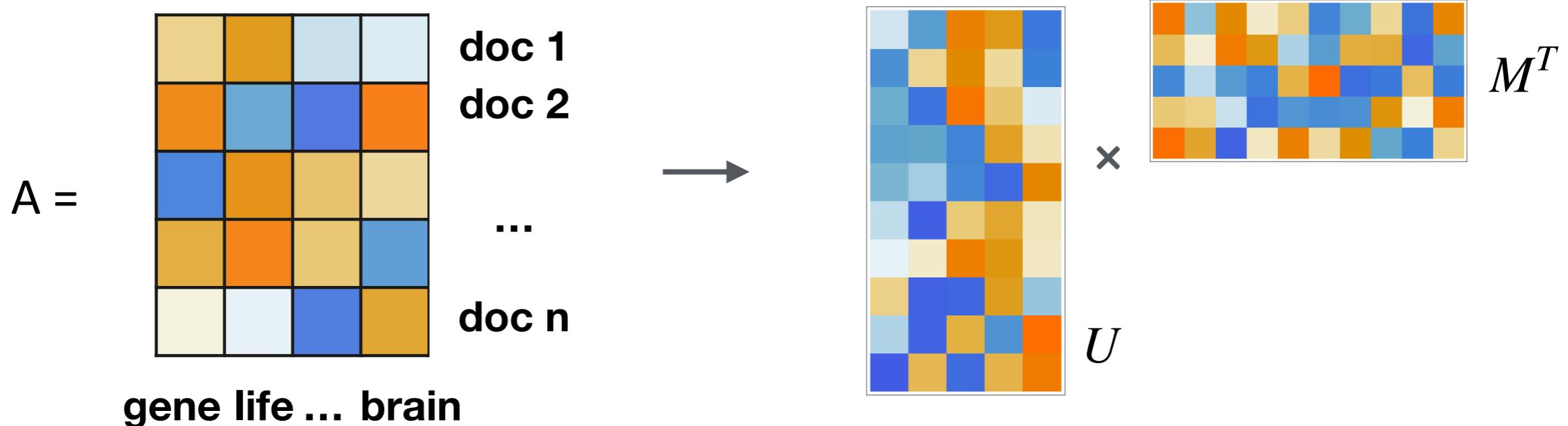
The word “gene” appears in doc 1 for A_{11} times because:

$$A_{11} = U_{1.}^T M_1.$$



Question: Explain the equation above in plain words based on the interpretation of topic and doc matrices.

Topic Model



Question: What is the difference between topic model and K-means you used in HW5-Problem 2?

Cluster 0 words:

america,written,san,ballet,wrote,medium,story,jersey,child,television,festival,band,appeared,series,began,p
oetry,writing,artist,museum,magazine,dance,writer,play,theatre,film

Cluster 1 words:

manifold,mit,scientist,earned,language,position,algebraic,applied,computing,contribution,analysis,student,a
dvanced,logic,problem,algorithm,geometry,ieee,paper,engineering,philosophy,mathematician,mathematical,compu
ter,mathematics

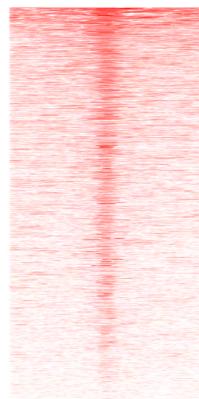
Cluster 2 words:

musical,album,concerto,electronic,artist,opera,studied,contemporary,record,london,concert,philharmonic,prem
iered,chamber,organ,conservatory,performed,recording,composition,hall,ensemble,symphony,composer,festival,o
rchestra

Topic Model

Applications to genomics: Doc = Lab, Word = Gene, Topic = Diseases

ChIP Data



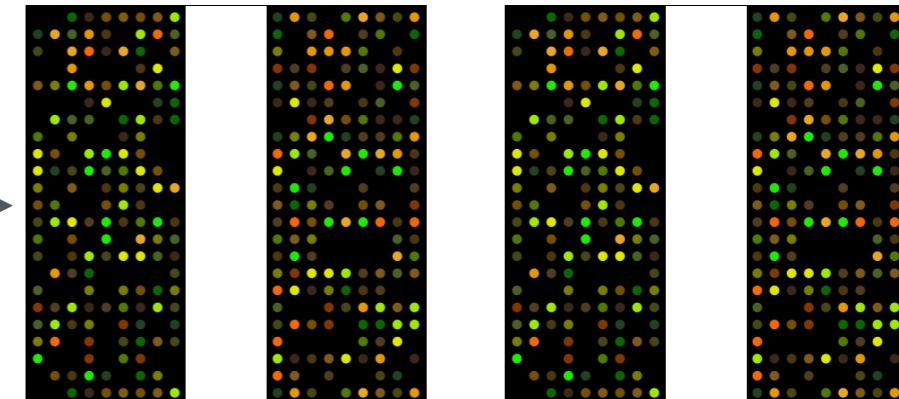
Genes (>12K)

BRCA1

MYC

SUZ12 ?

Gene Expression



Transcription
Factor (>1K)

Lab 1 Lab 2 . . .

Sample Size >10K

Ewing's
Sarcoma

Breast
Cancer

Biological
Context (>2K)



Match



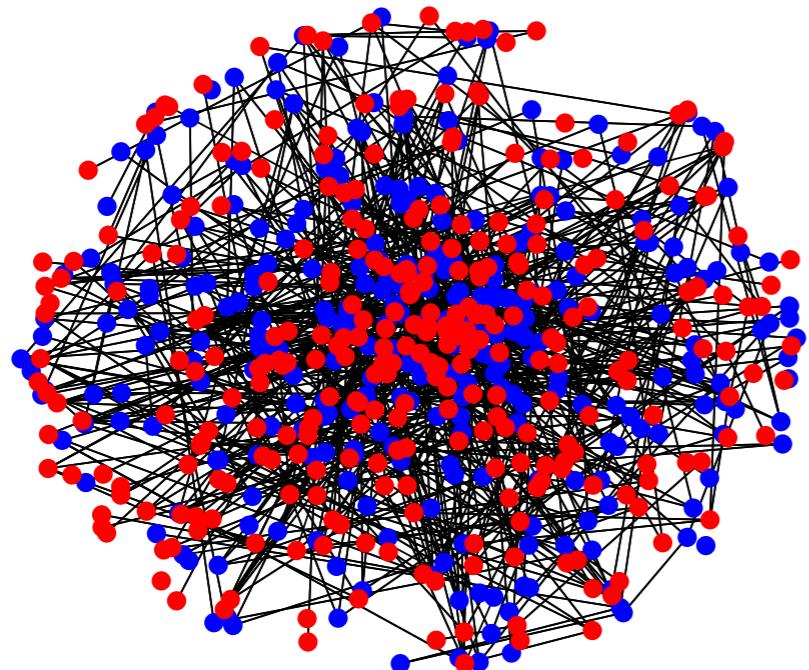
Regulatory Association

Principal Components

Social Network

Community detection for social network:

Facebook friend network/citation network/gene network/brain network



Network you think

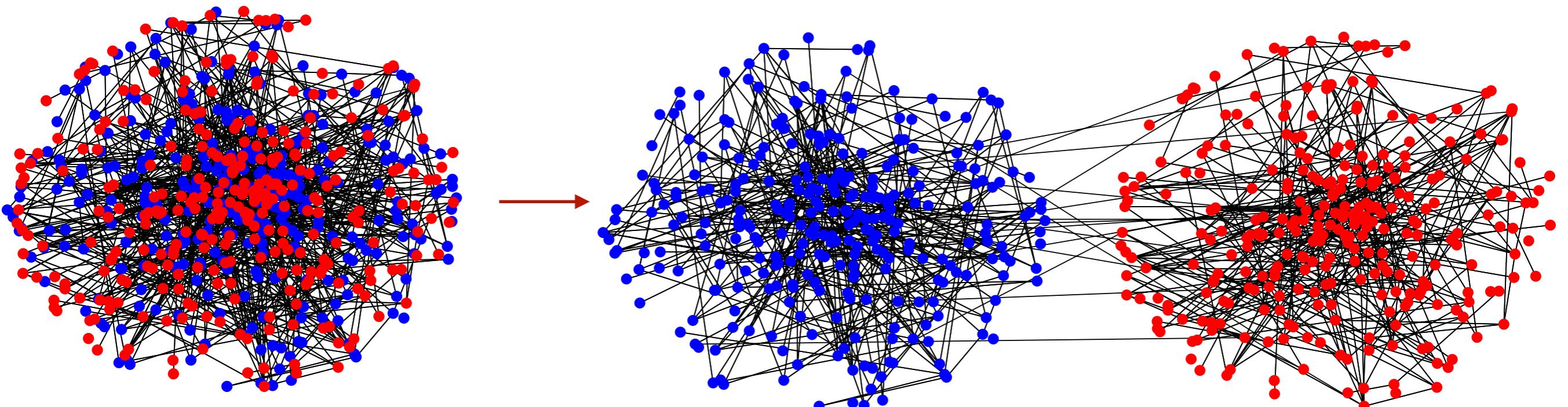


Network in reality

Social Network

Community detection for social network:

Goal: Cluster the nodes in different communities/groups



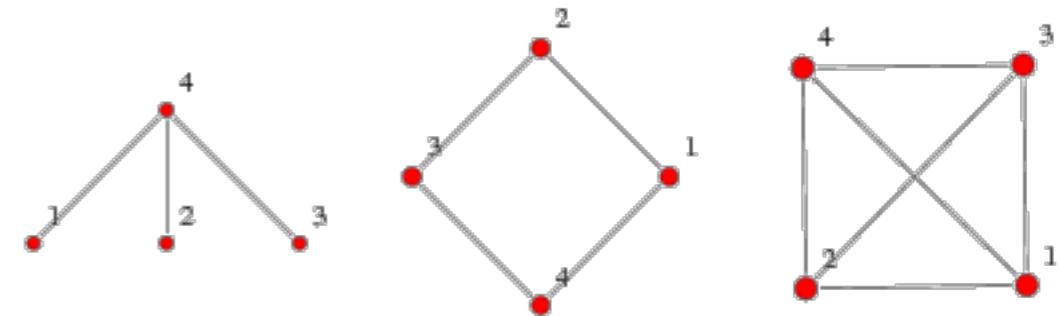
Community Detection

Goal: Cluster the nodes in different groups

Adjacency matrix of graph with n nodes

$$A \in \mathbb{R}^{n \times n}$$

$$A_{ij} = \begin{cases} 1 & \text{nodes } i, j \text{ are connected} \\ 0 & \text{otherwise} \end{cases}$$

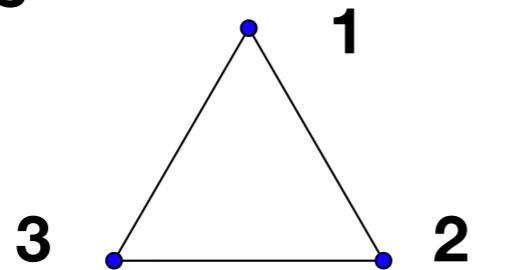


$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

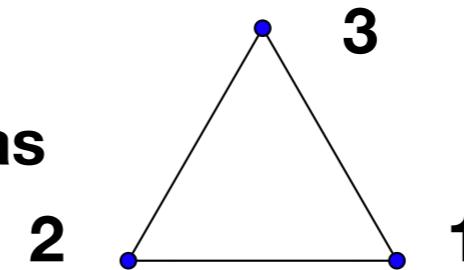
$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Adjacency matrix represents the same graph even if we **permute** the node labels



is the same as

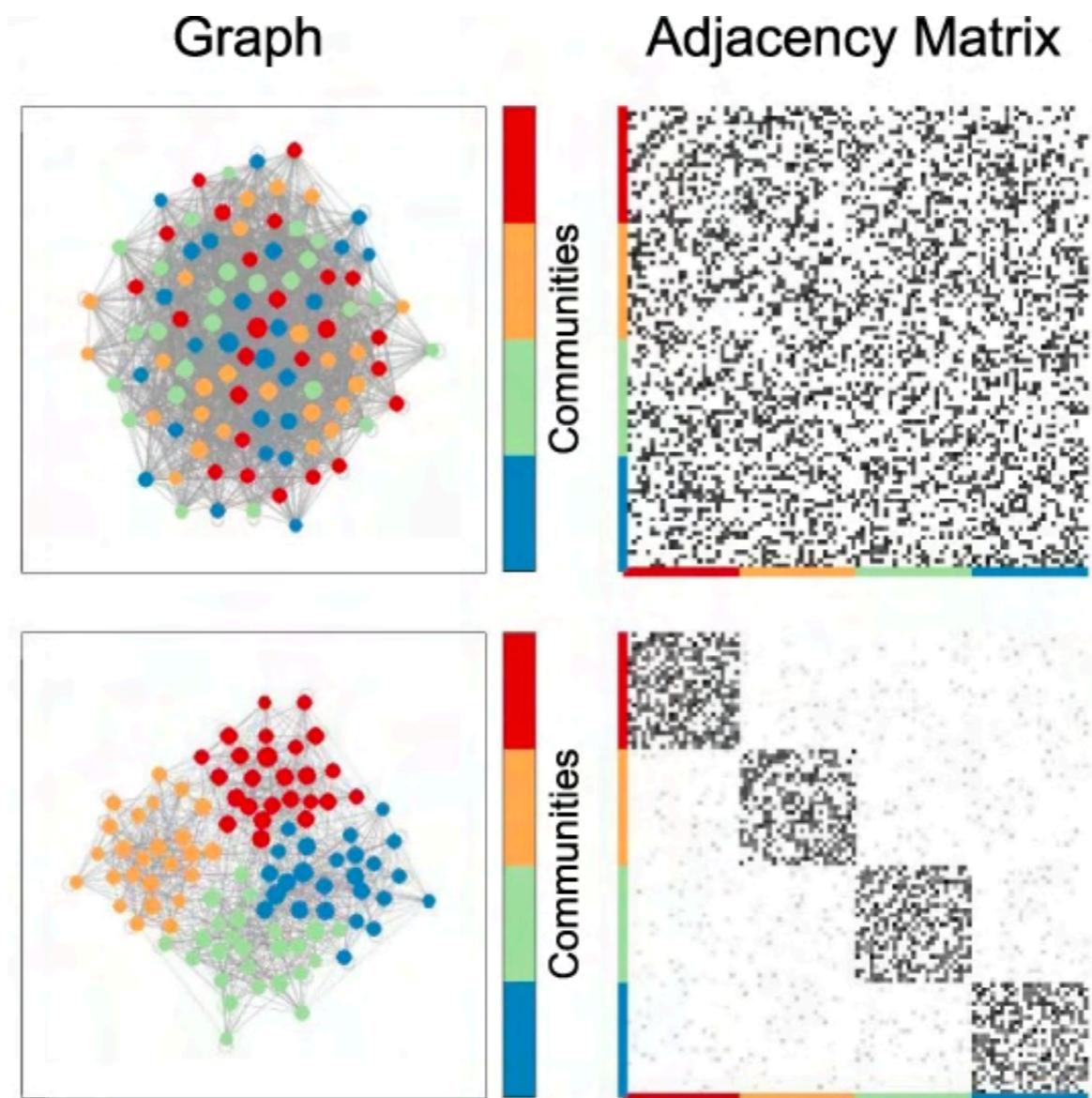


Question: Can we use K-means or EM for community detection? How?

Community Detection

Goal: Cluster the nodes in different groups

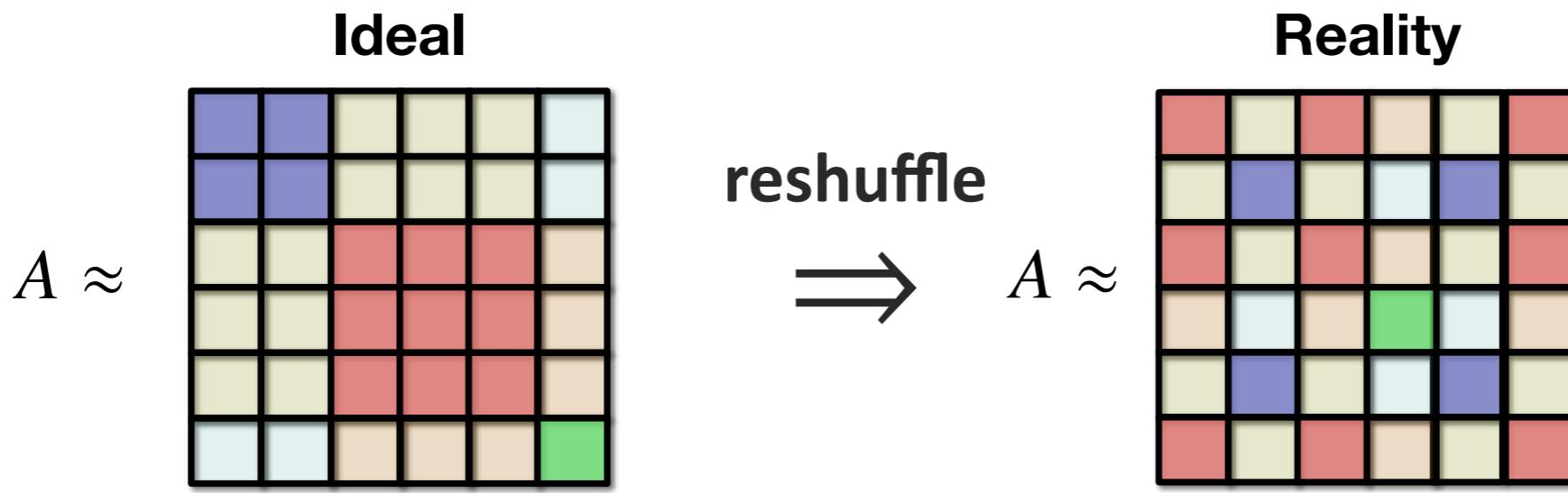
Idea: Edges are more likely to be connected within community than between community



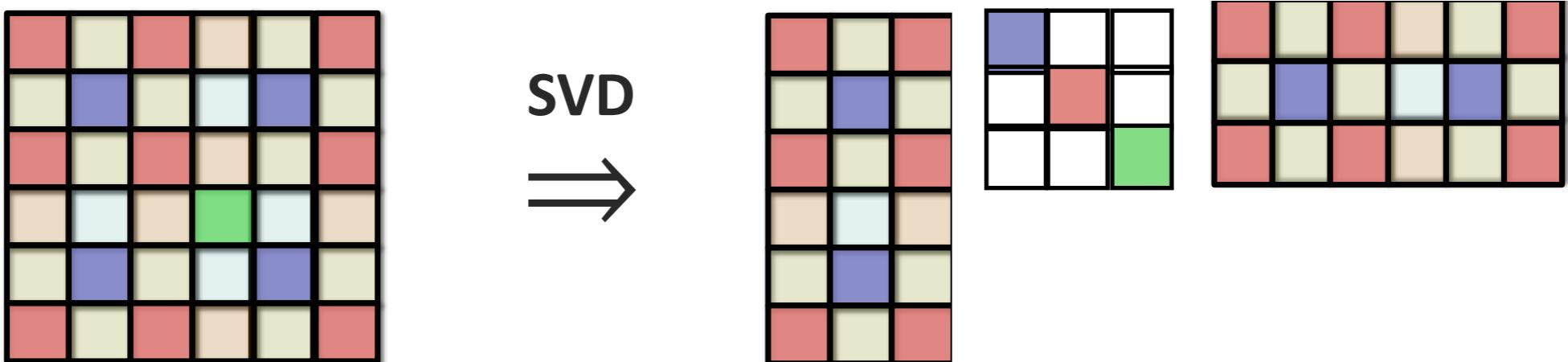
Community Detection

Goal: Cluster the nodes in different groups

Idea: Edges are more likely to be connected within community than between community

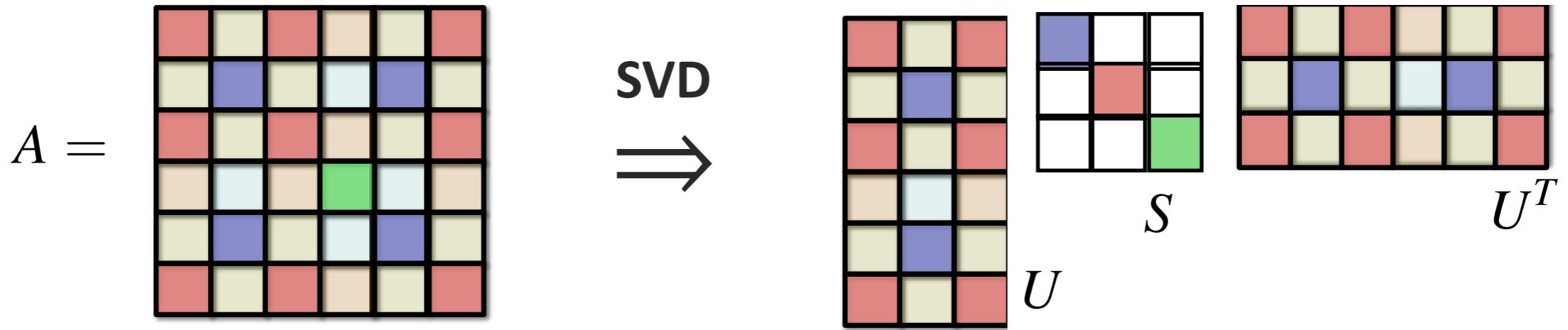


Step 1. SVD $A = USU^T$ (A is symmetric, so $V = U$)

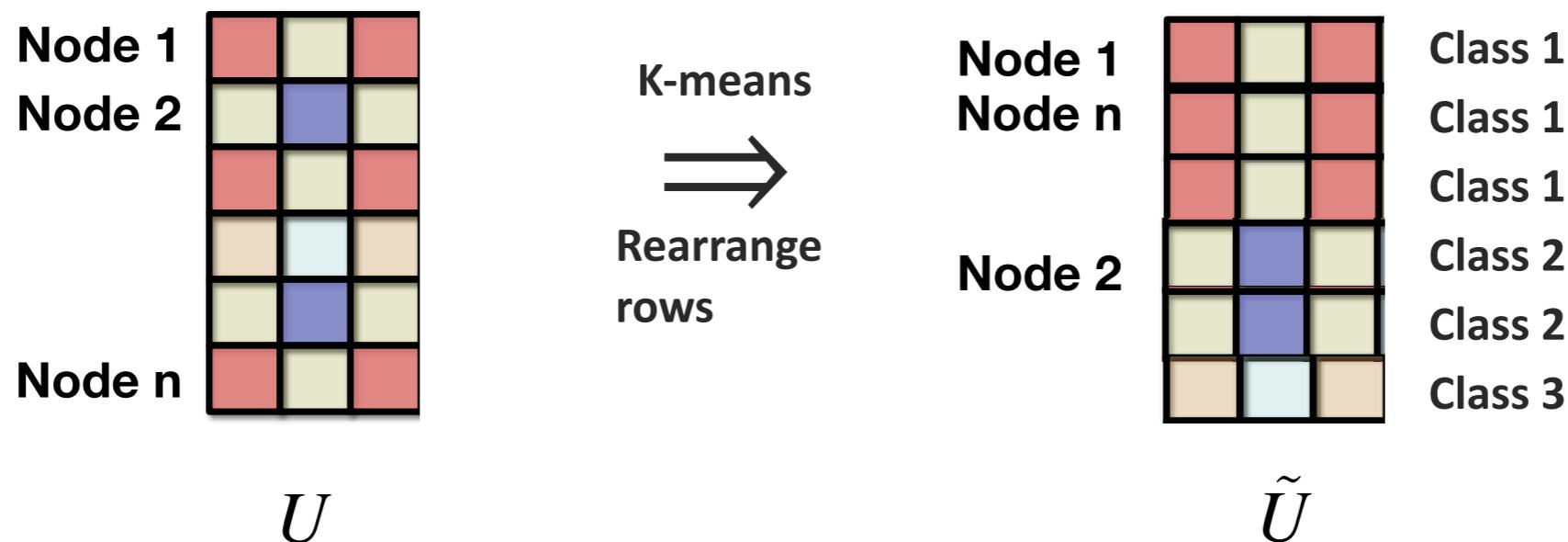


Community Detection

Step 1. SVD $A = USU^T$ (A is symmetric, so $V = U$)

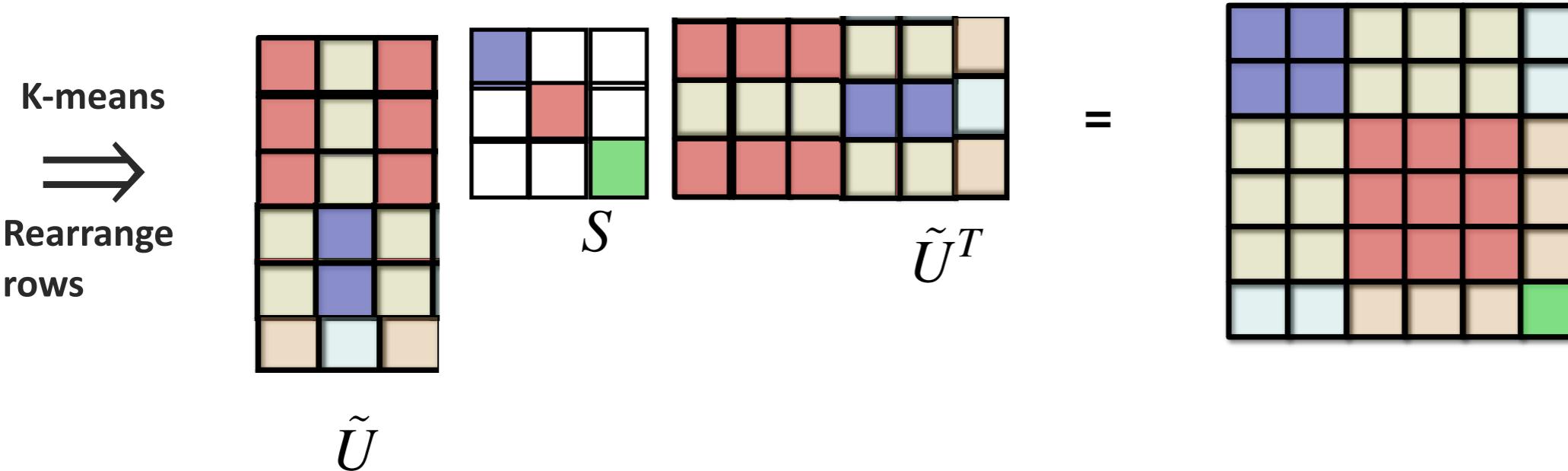
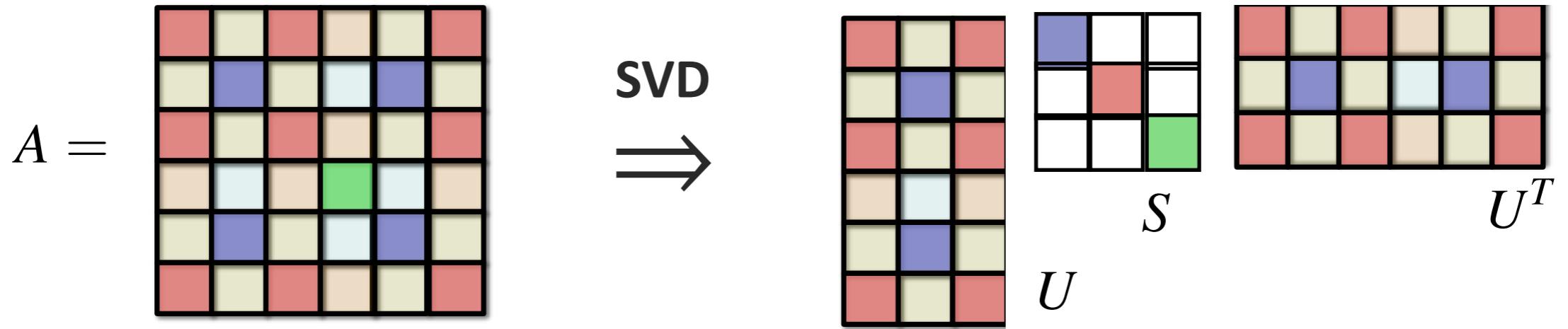


Step 2. Clustering the rows of U , rearrange the rows by the cluster groups



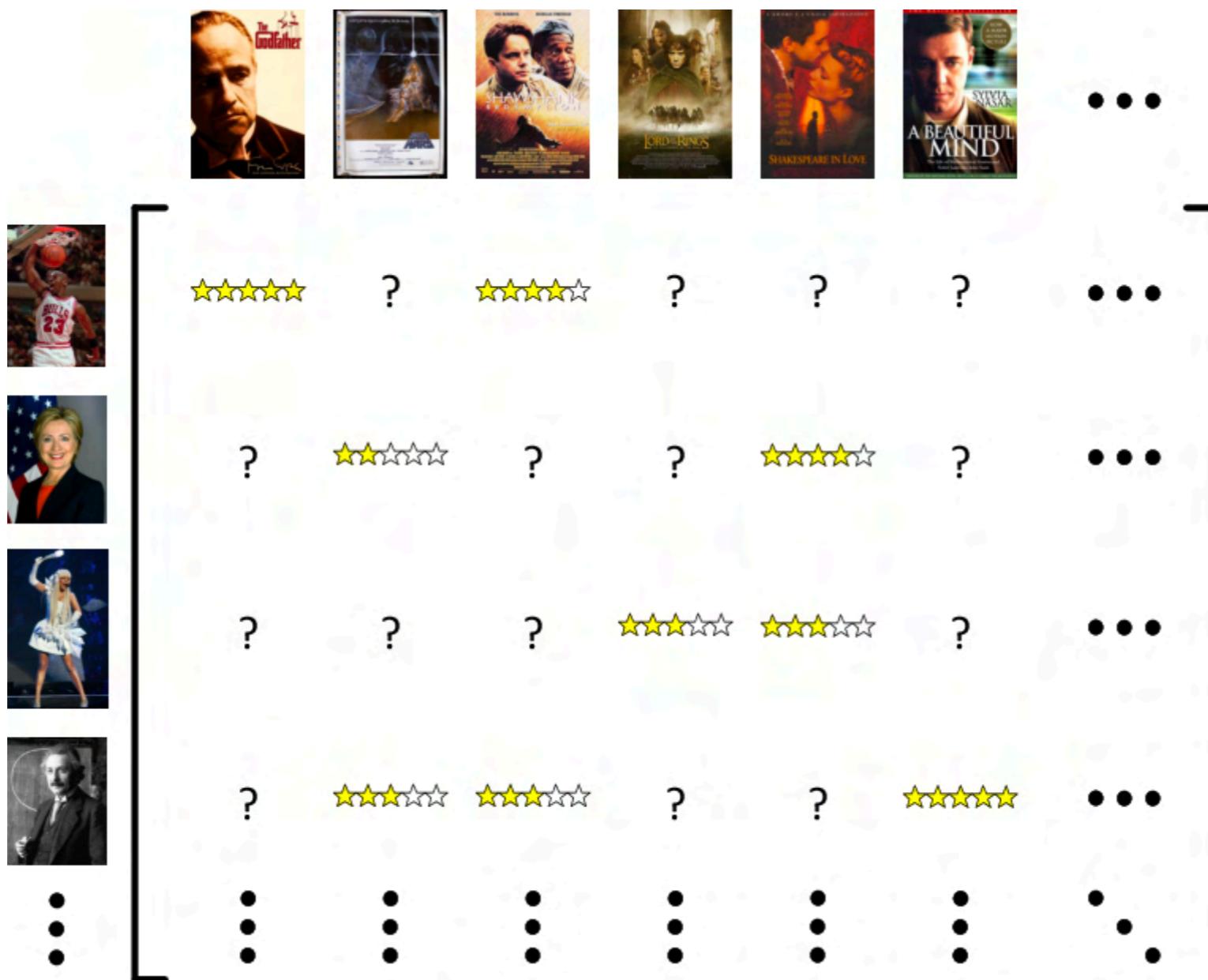
Community Detection

We can get a more aligned adjacency matrix



Recommendation System

Netflix Prize: Predict the rating of movies, \$1 million for accuracy > 90%

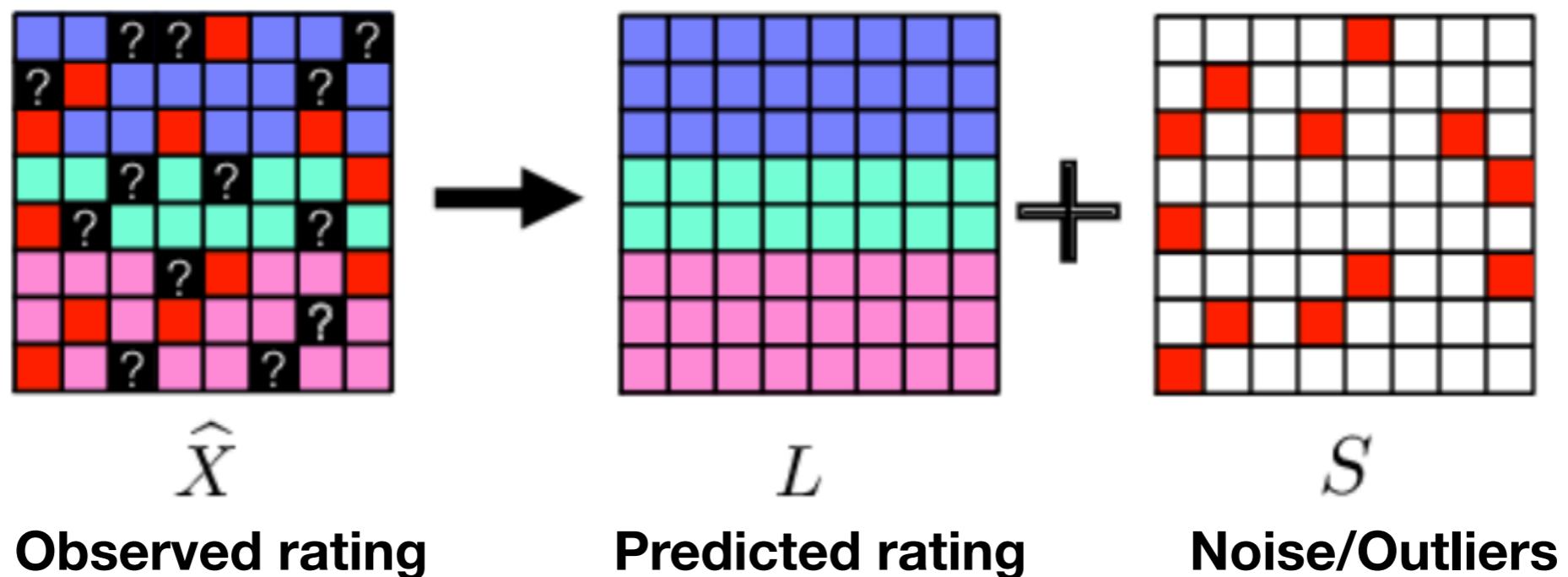


Similar problems in amazon, youtube, health care...

Recommendation System

Idea: Similar people have similar taste of movies

Translate in math language: rating matrix is close to low rank



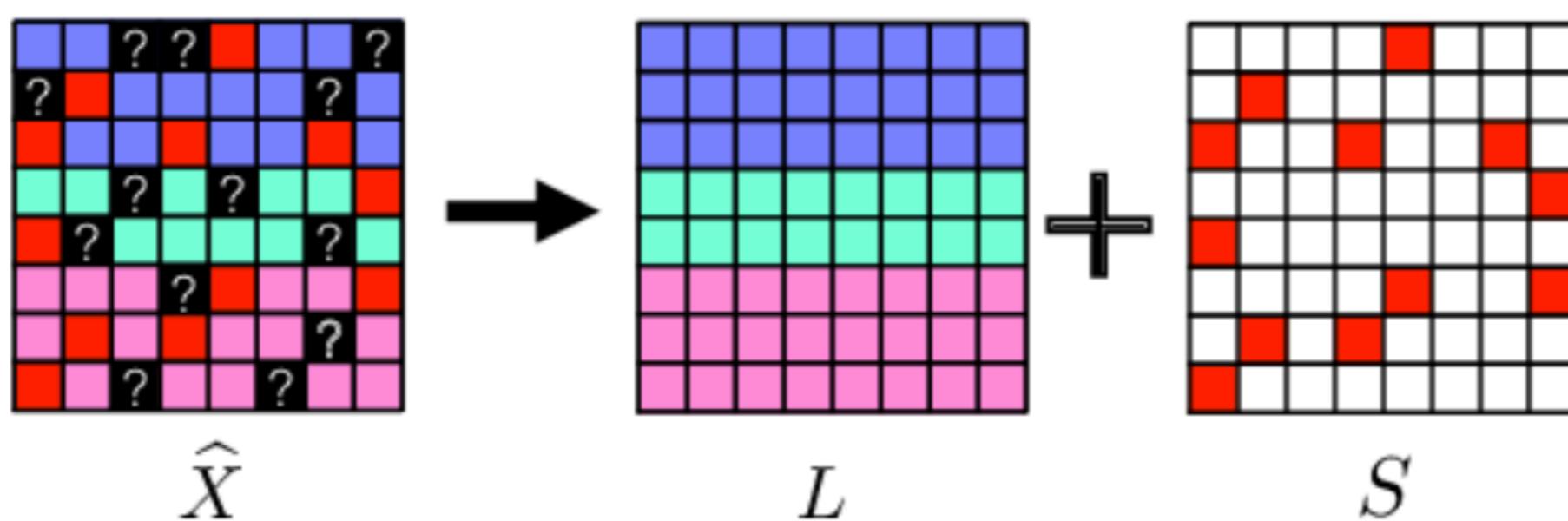
Step 1. SVD $\hat{X} = U S V^T$

Step 2. Choose top k , $L = U_k S_k V_k^T$

Better (but more sophisticated method) can be found in <https://arxiv.org/abs/0805.4471>

Recommendation System

Application to computer vision: How to remove background



Recommendation System

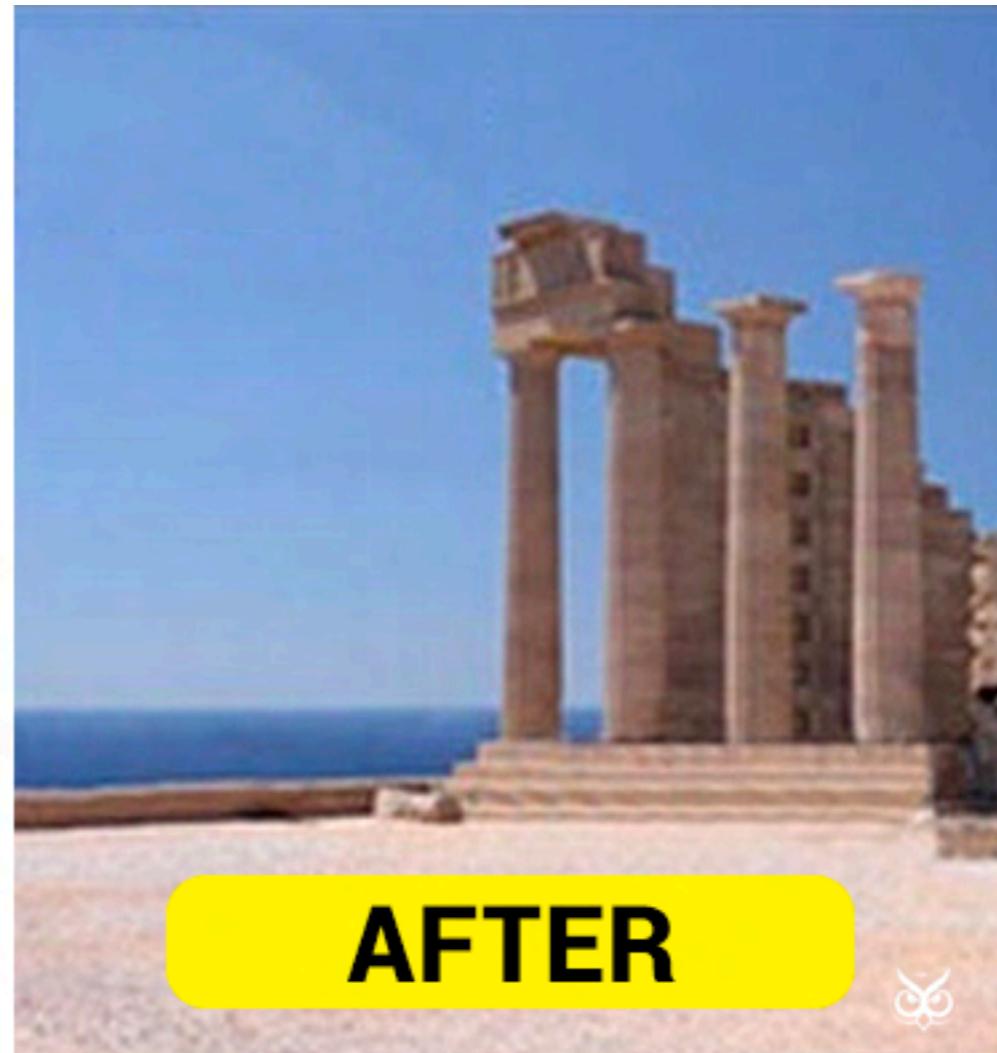


Question: How to remove annoying tourists?



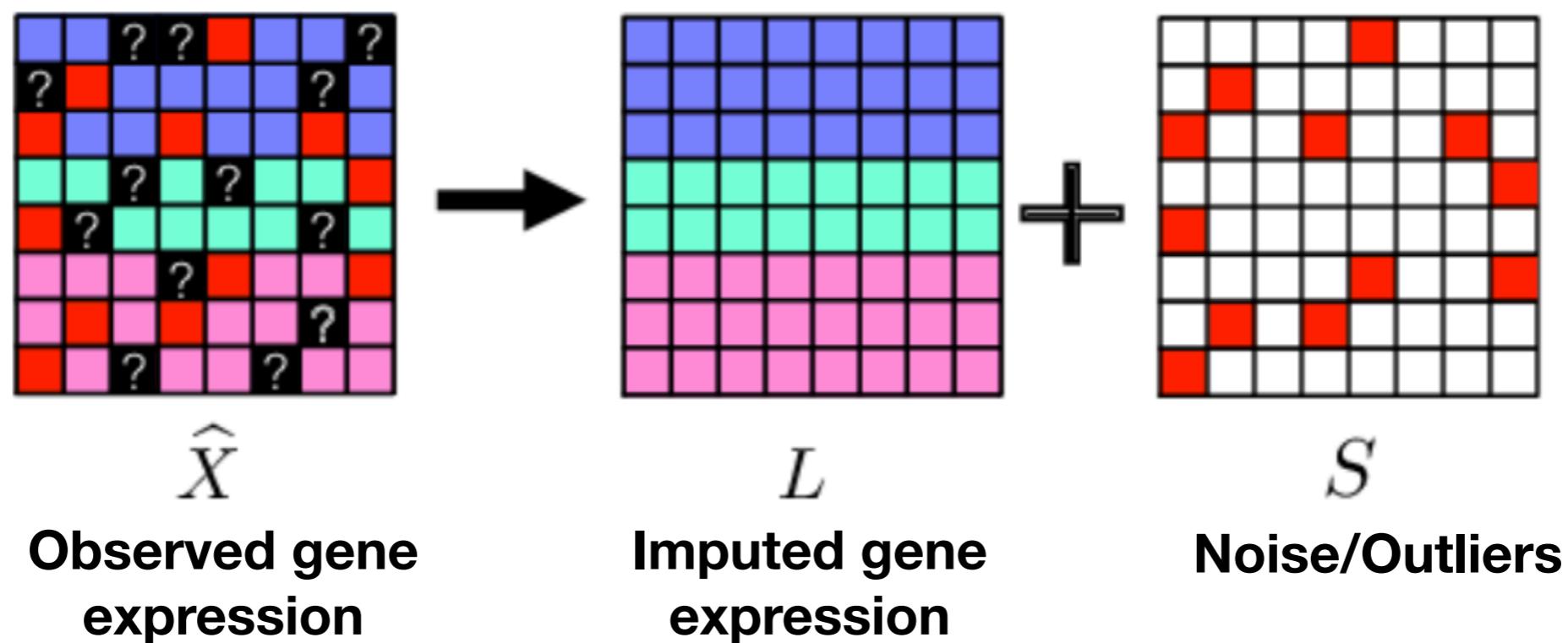
Recommendation System

Answer: Take a few shots and median is faster than PCA (sometimes even better)



Recommendation System

Application to genomic data: Imputing the missing observations



Boosting

Junwei Lu



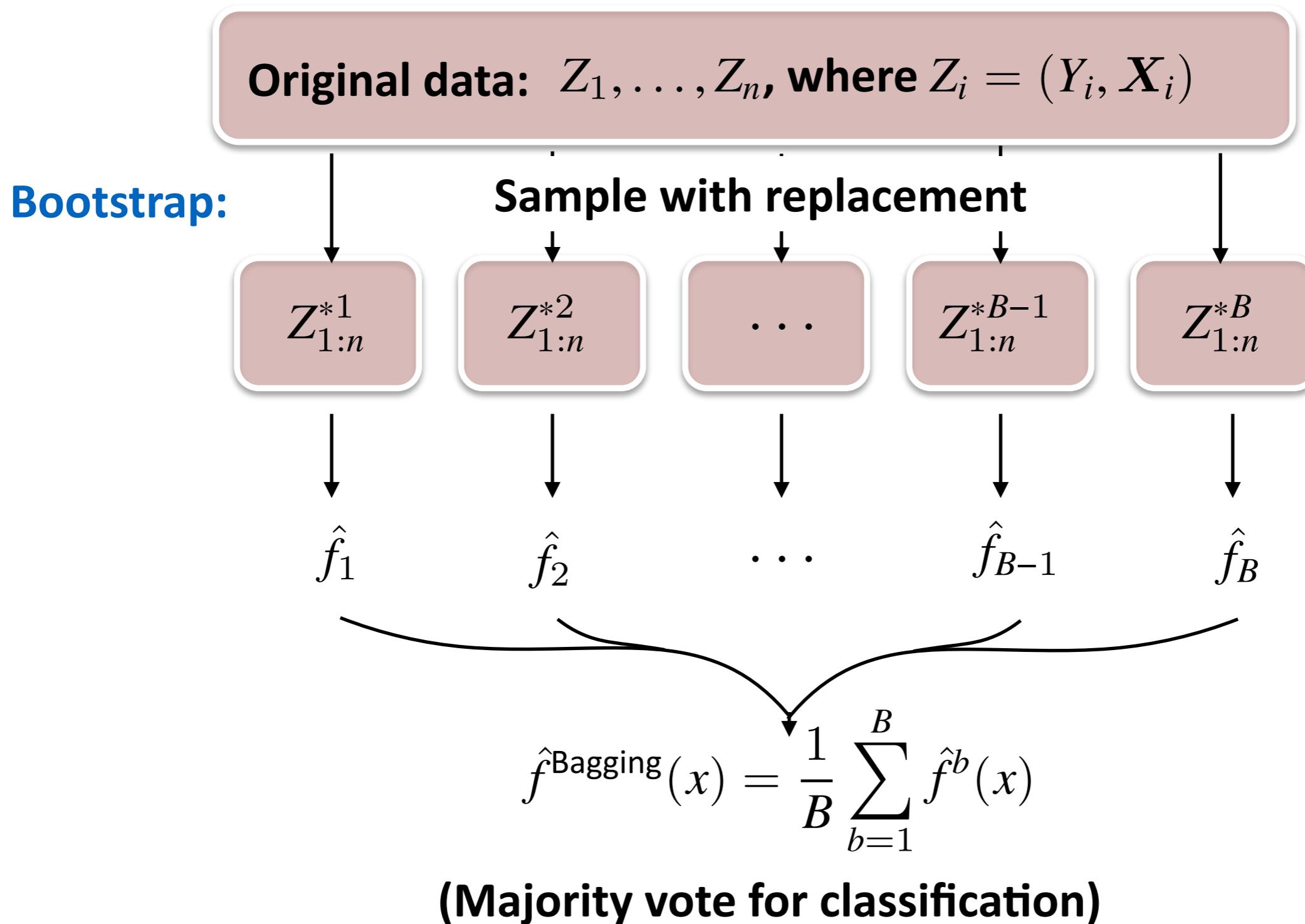
HARVARD
T.H. CHAN

SCHOOL OF PUBLIC HEALTH
Department of Biostatistics



Review of Bagging

Main idea: Control the variance of model via resampling



Boosting

Boosting is another way of combining models into an ensemble.

Like bagging, boosting can be applied to any model class.

Basic idea: Sequentially train **simple** models to slowly reduce the **residual error** in a targeted way.

Boosting Algorithm

Algorithm:

Set $\hat{f}(x) = 0$ and $r_i = y_i$ **for all** $i = 1, \dots, n$

For $b = 1, 2, \dots, B$ (Tree, linear regression, etc)



(a) Fit $\hat{f}^b(x)$ **via a given method using the training data** $\{(r_i, x_i)\}_{i=1}^n$

(b) Update estimator \hat{f} : $\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$

(c) Update the residuals: $r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$

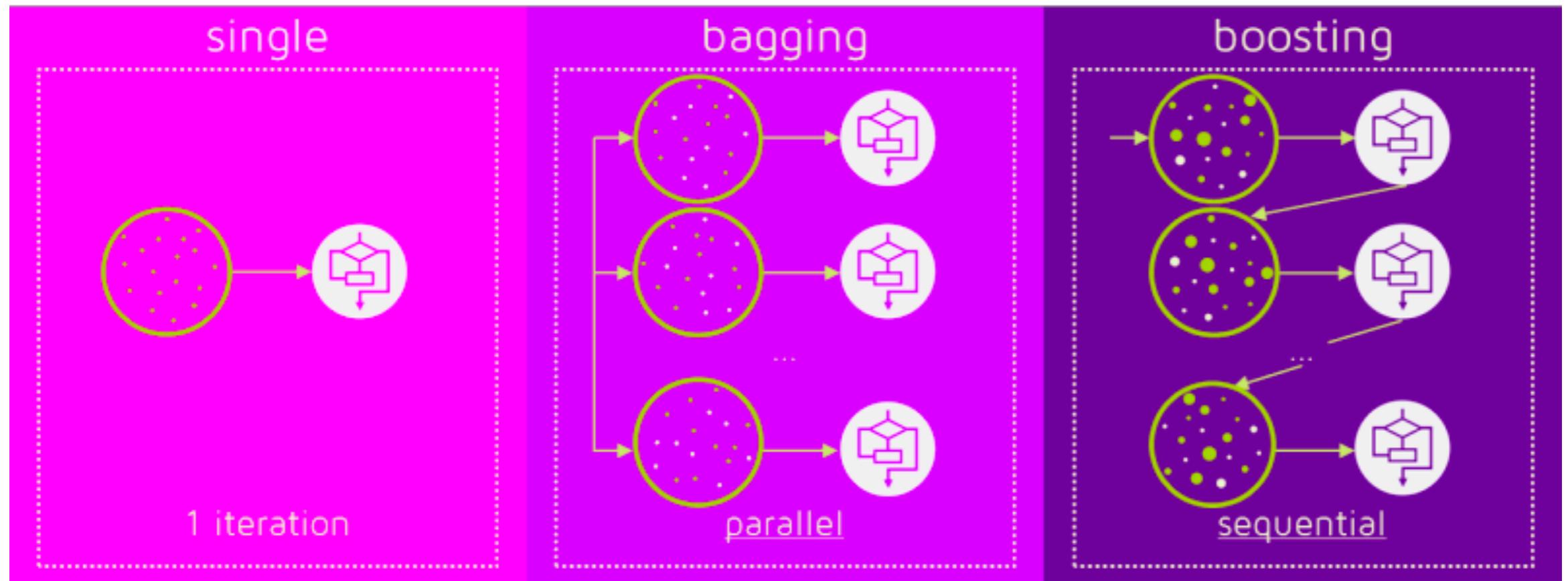
Output the boosted model:

$$\hat{f}(x) = \sum_{b=1}^B \lambda \hat{f}^b(x)$$

Boosting vs Bagging

- **Bagging constructs decorrelated prediction models**
- **In boosting, each model depends on others**

Boosting vs Bagging



General Boosting

Let $L(y, \hat{y})$ be a loss function, e.g., $L(y, \hat{y}) = (y - \hat{y})^2$

Algorithm:

$$\hat{f}_0(x) = \operatorname{argmin}_{h \in \mathcal{H}} \sum_{i=1}^n L(y_i, h)$$

For $b = 1, 2, \dots, B$

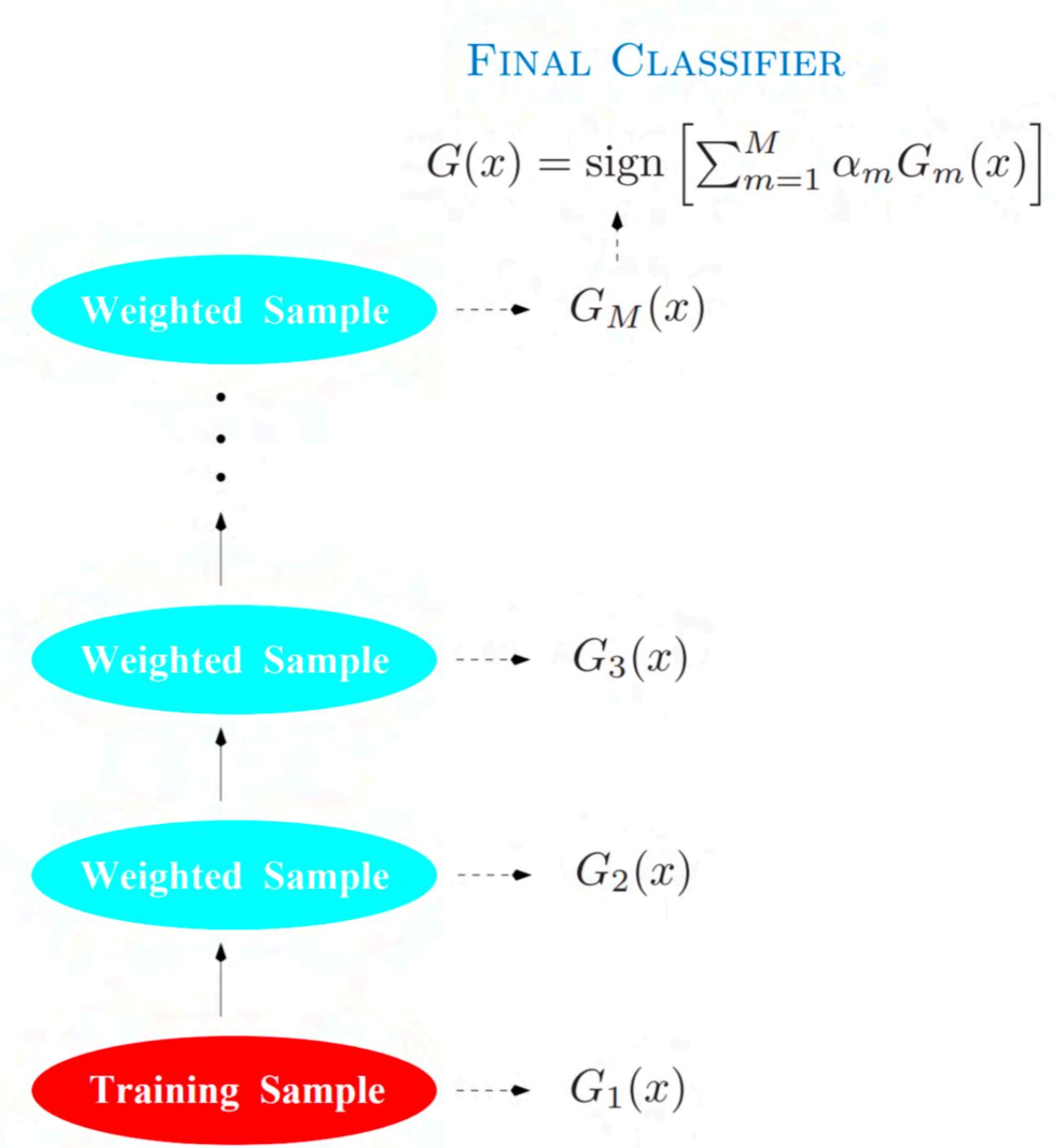
$$\hat{f}_b(x) = \operatorname{argmin}_{h_b \in \mathcal{H}} \sum_{i=1}^n L(y_i, \sum_{m=0}^{b-1} \hat{f}_m(x_i) + h_b(x_i))$$

When $L(y, \hat{y}) = (y - \hat{y})^2$,

$$\hat{f}_b(x) = \operatorname{argmin}_{h_b \in \mathcal{H}} \sum_{i=1}^n \left(y_i - \sum_{m=0}^{b-1} \hat{f}_m(x_i) - h_b(x_i) \right)^2 = \operatorname{argmin}_{h_b \in \mathcal{H}} \sum_{i=1}^n (r_i - h_b(x_i))^2$$

Boosting for Binary Classification

Assume $y \in \{-1, 1\}$ and $h(x) \in \{-1, 1\}$ is a binary classifier.



Boosting for Binary Classification

Assume $y \in \{-1, 1\}$ **and** $h(x) \in \{-1, 1\}$ **is a binary classifier.**

Consider the loss $L(y, \hat{f}(x)) = \exp(-y\hat{f}(x))$ **(exponential loss)**

We have

$$\begin{aligned} L(y_i, \hat{f}(x_i) + \beta h(x_i)) &= \exp(-y_i \hat{f}(x_i) - y_i \beta h(x_i)) \\ &= w_i \exp(-\beta y_i h(x_i)) \end{aligned}$$

where $w_i = \exp(-y_i \hat{f}(x_i))$ **are weights.**

It turns out that $\sum_i w_i \exp(-\beta y_i h(x_i))$ **is minimized by**

$$h = \operatorname{argmin}_{h \in \mathcal{H}} \sum_i w_i \mathbb{I}(y_i \neq h(x_i))$$

$$\beta = \frac{1}{2} \log \left(\frac{\sum_i w_i \mathbb{I}(y_i = h(x_i))}{\sum_i w_i \mathbb{I}(y_i \neq h(x_i))} \right).$$

Boosting for Binary Classification

Assume $y \in \{-1, 1\}$ **We estimate the classifier**

$$\hat{y} = \text{sign}(\hat{f}(x)) = \text{sign}(\sum_b \beta_b h_b(x))$$

Algorithm: AdaBoost

$$w_i = 1/n$$

For $b = 1, 2, \dots, B$

$$h_b = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \sum_i w_i \mathbf{I}(y_i \neq h(x_i)) \quad \text{(classification err)}$$

$$\beta_b = \frac{1}{2} \log \left(\frac{\sum_i w_i \mathbf{I}(y_i = h(x_i))}{\sum_i w_i \mathbf{I}(y_i \neq h(x_i))} \right). \quad \text{(readjust the weights)}$$

$$\hat{f}_b(x) = \sum_b \beta_b h_b(x)$$

$$w_i = \exp(-y_i \hat{f}_b(x_i)) \quad \text{(update the weights)}$$

Boosting for Binary Classification

Assume $y \in \{-1, 1\}$ **We estimate the classifier**

$$\hat{y} = \text{sign}(\hat{f}(x)) = \text{sign}(\sum_b \beta_b h_b(x))$$

Algorithm: AdaBoost

$$w_i = 1/n$$

For $b = 1, 2, \dots, B$

$$h_b = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \sum_i w_i \mathbb{I}(y_i \neq h(x_i)) \longrightarrow \begin{array}{l} h \in \mathcal{H} \\ \text{Simple classifier like linear} \end{array}$$

$$\beta_b = \frac{1}{2} \log \left(\frac{\sum_i w_i \mathbb{I}(y_i = h(x_i))}{\sum_i w_i \mathbb{I}(y_i \neq h(x_i))} \right).$$

$$\hat{f}_b(x) = \sum_b \beta_b h_b(x)$$

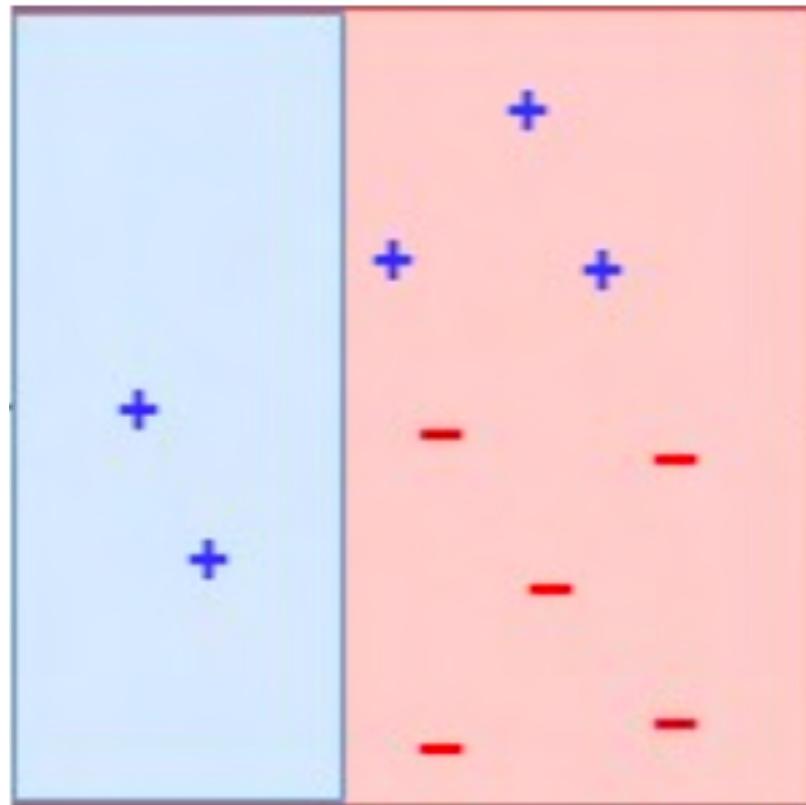
$$w_i = \exp(-y_i \hat{f}_b(x_i))$$

Example of AdaBoost

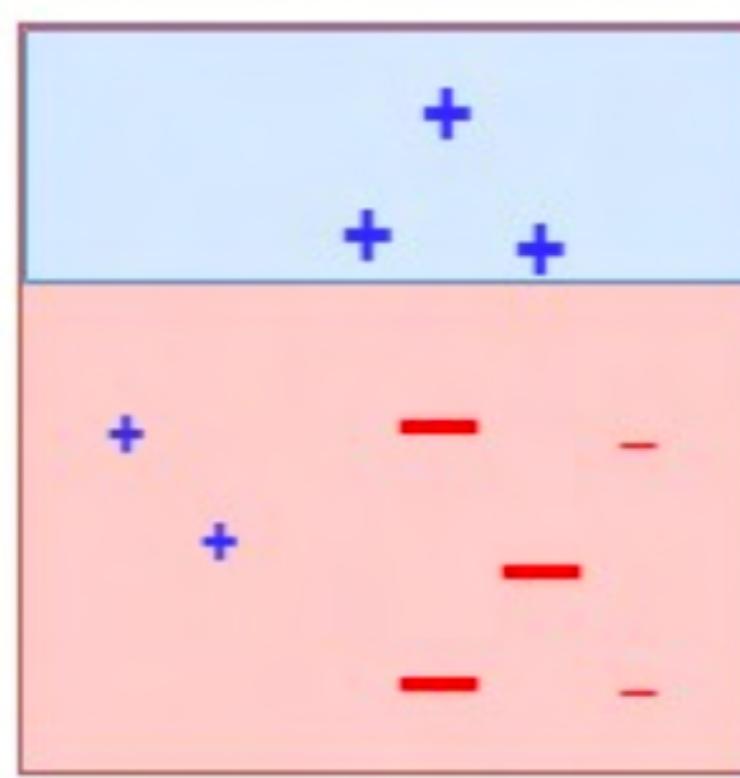
$h \in \mathcal{H}$ Simple classifiers: Only vertical or horizontal lines

$$h_b = \operatorname{argmin}_{h \in \mathcal{H}} \sum_i w_i \mathbb{I}(y_i \neq h(x_i))$$

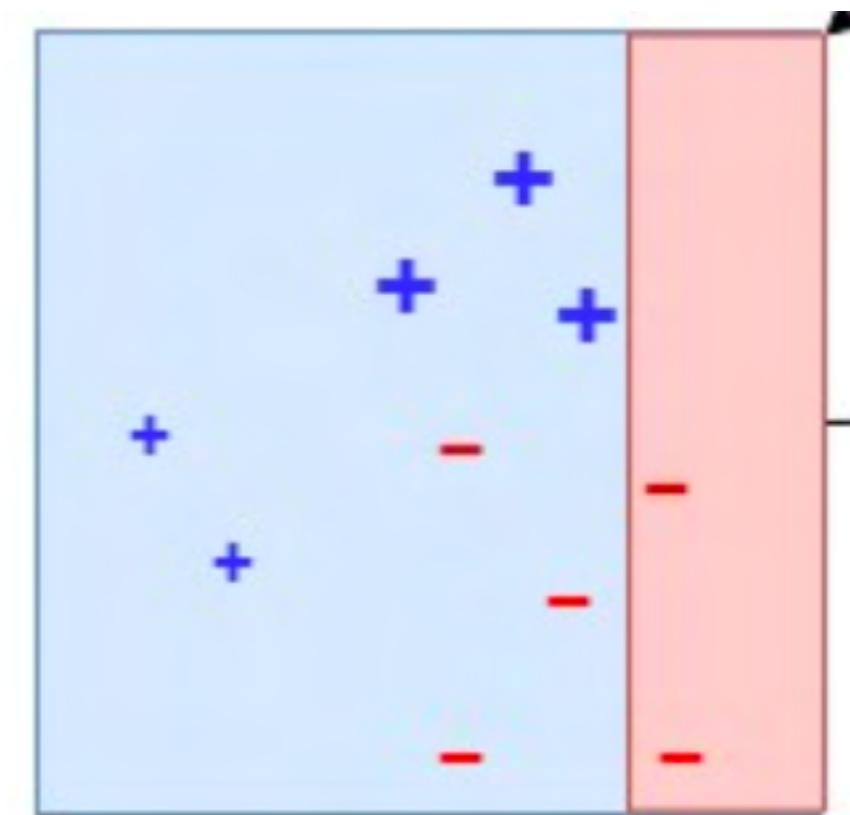
$$\beta_b = \frac{1}{2} \log \left(\frac{\sum_i w_i \mathbb{I}(y_i = h(x_i))}{\sum_i w_i \mathbb{I}(y_i \neq h(x_i))} \right).$$



b=1



b=2



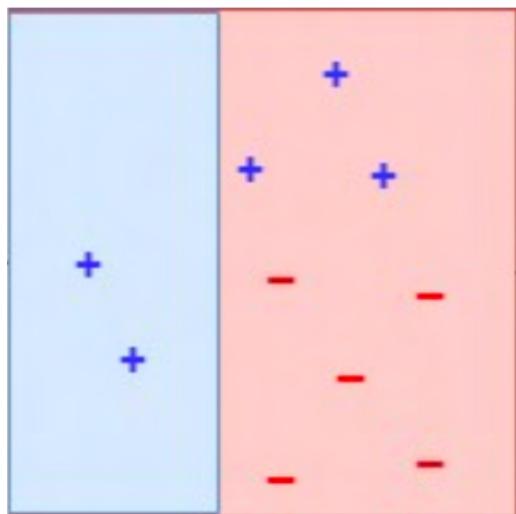
b=3

Example of AdaBoost

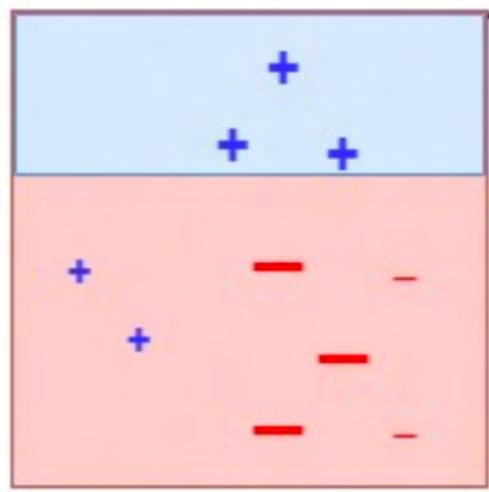
$h \in \mathcal{H}$ Simple classifiers: Only vertical or horizontal lines

$$h_b = \operatorname{argmin}_{h \in \mathcal{H}} \sum_i w_i \mathbb{I}(y_i \neq h(x_i))$$

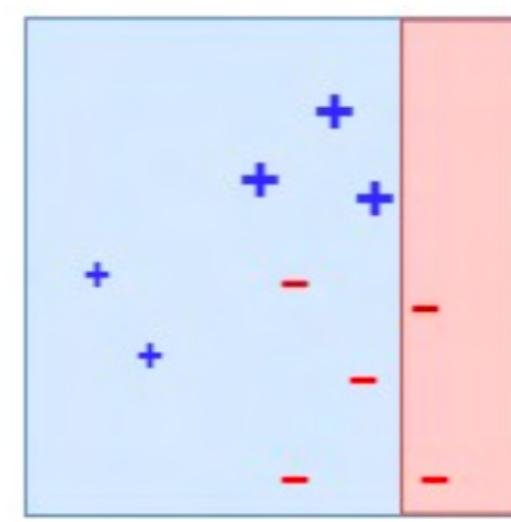
$$\beta_b = \frac{1}{2} \log \left(\frac{\sum_i w_i \mathbb{I}(y_i = h(x_i))}{\sum_i w_i \mathbb{I}(y_i \neq h(x_i))} \right).$$



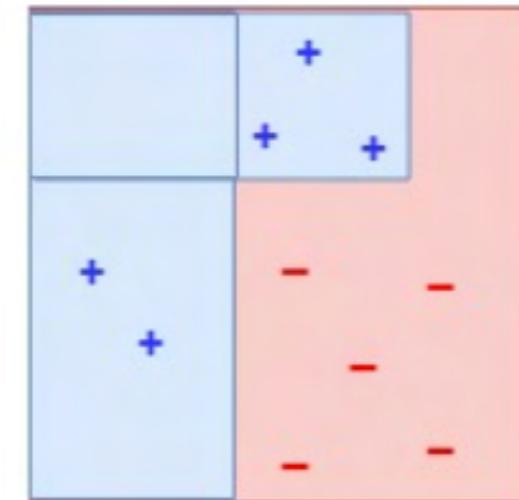
b=1



b=2



b=3



$\operatorname{sign}(\sum_b \beta_b h_b(x))$

AdaBoost

Algorithm: AdaBoost

$$w_i = 1/n$$

For $b = 1, 2, \dots, B$

$$h_b = \operatorname{argmin}_{h \in \mathcal{H}} \sum_i w_i \mathbb{I}(y_i \neq h(x_i))$$

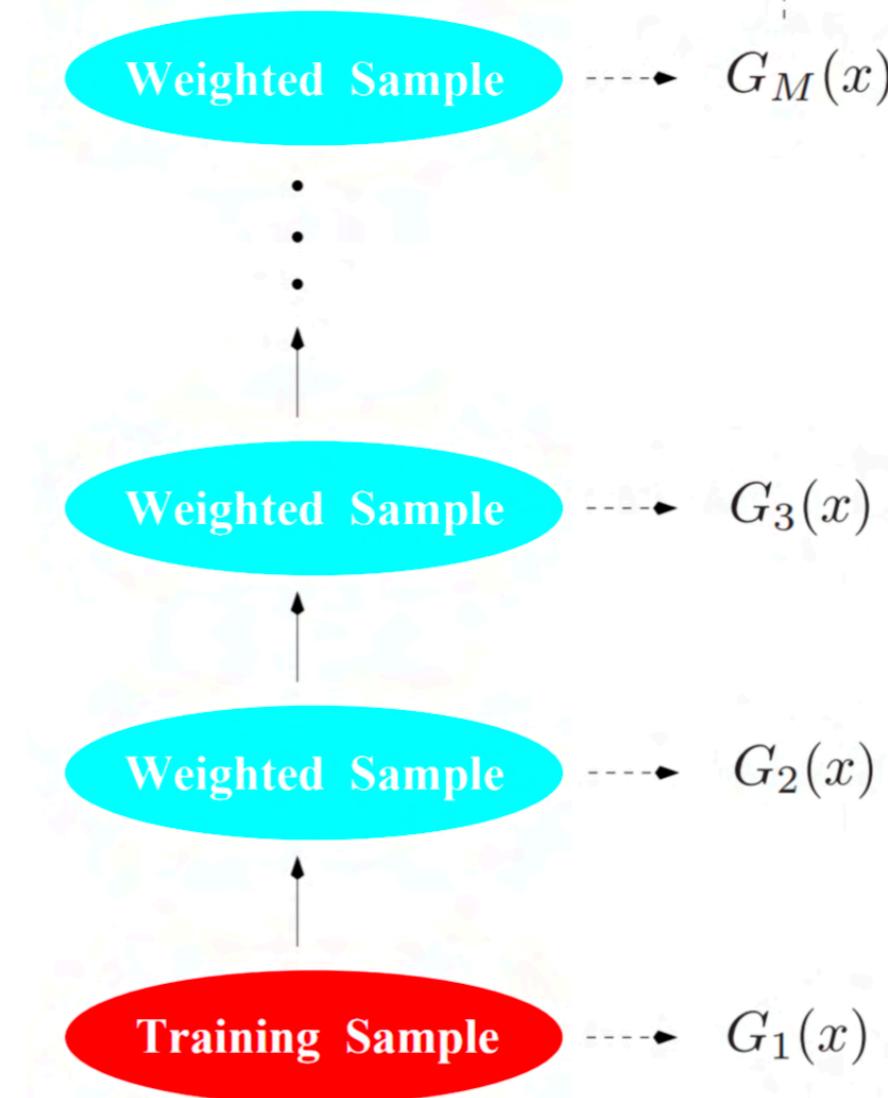
$$\beta_b = \frac{1}{2} \log \left(\frac{\sum_i w_i \mathbb{I}(y_i = h(x_i))}{\sum_i w_i \mathbb{I}(y_i \neq h(x_i))} \right).$$

$$\hat{f}_b(x) = \sum_b \beta_b h_b(x)$$

$$w_i = \exp(-y_i \hat{f}_b(x_i))$$

FINAL CLASSIFIER

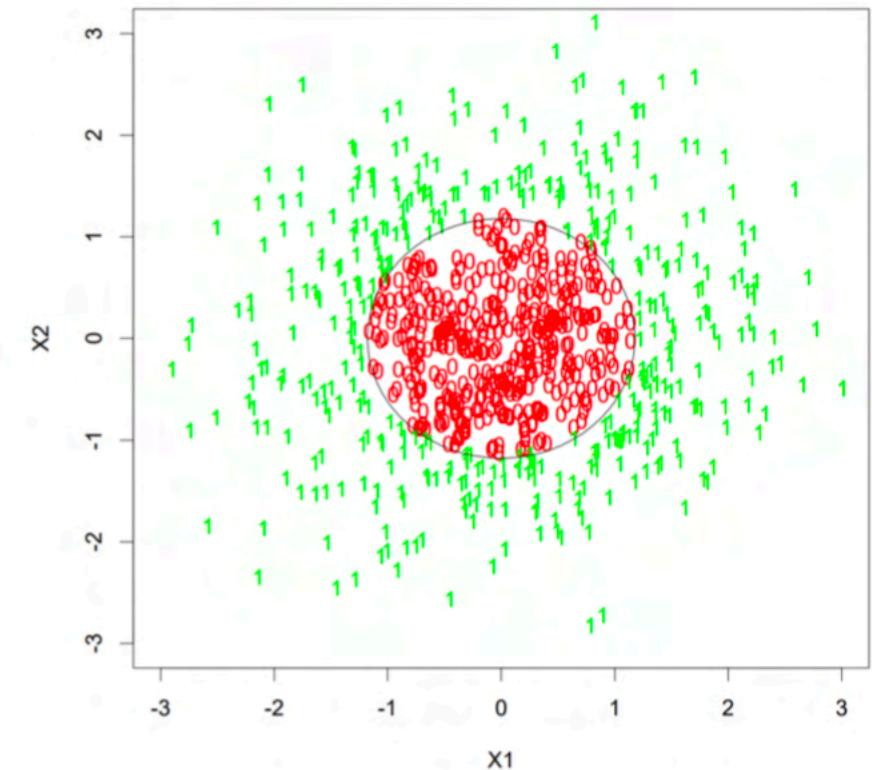
$$G(x) = \operatorname{sign} \left[\sum_{m=1}^M \alpha_m G_m(x) \right]$$



Nested spheres example

- Simulated binary classification problem.
- Predictors: $X_1, \dots, X_p \sim \mathcal{N}(0, 1)$ i.i.d.
- Outcome:

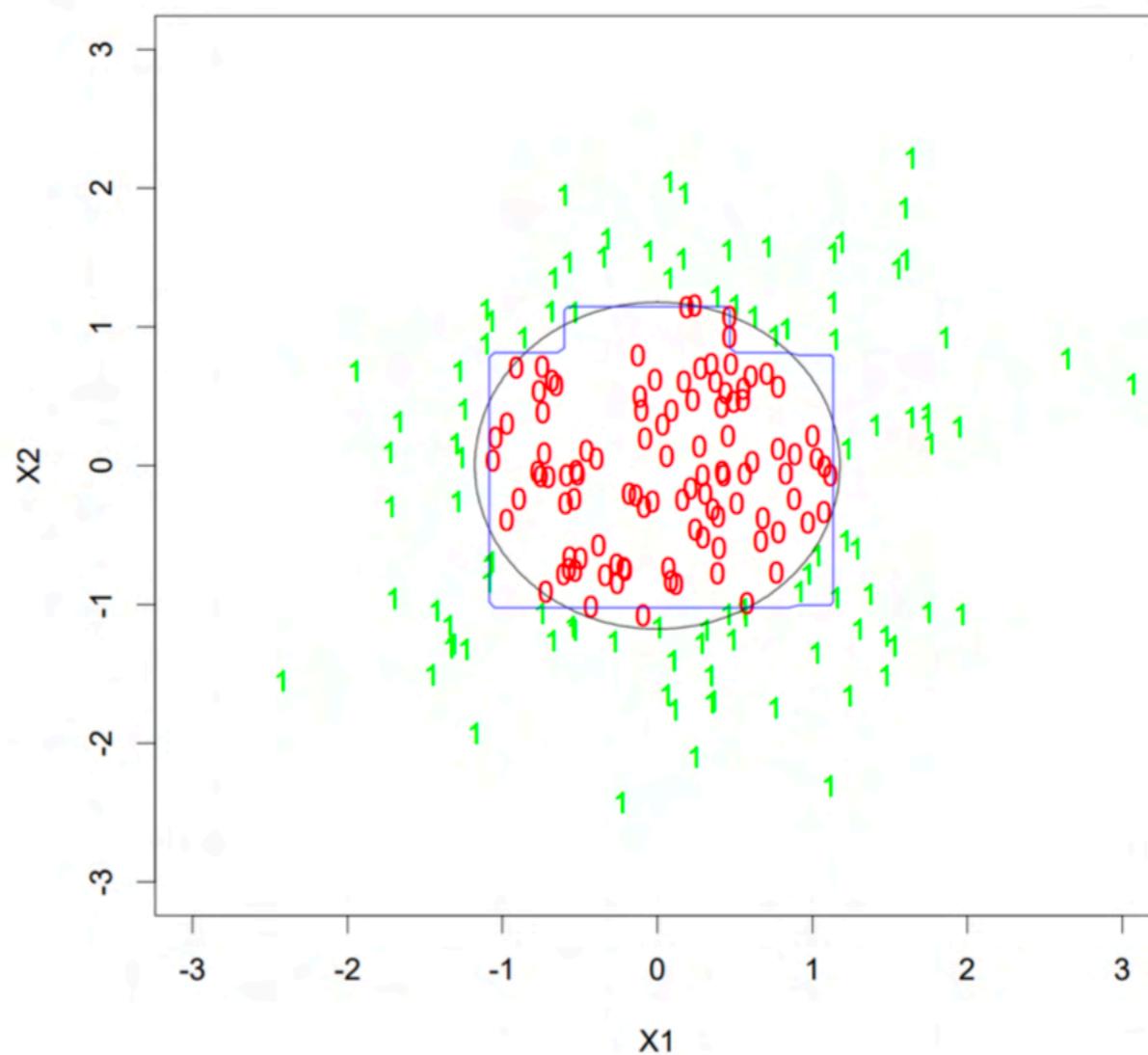
$$Y = \begin{cases} 1 & \text{if } \sum_{j=1}^p X_j^2 > c \\ -1 & \text{if otherwise.} \end{cases}$$



- $c = \text{median}(\sum_{j=1}^p X_j^2)$, roughly the same number of points from each class
- (What is the error rate of the Bayes optimal classifier?)

Nested spheres example

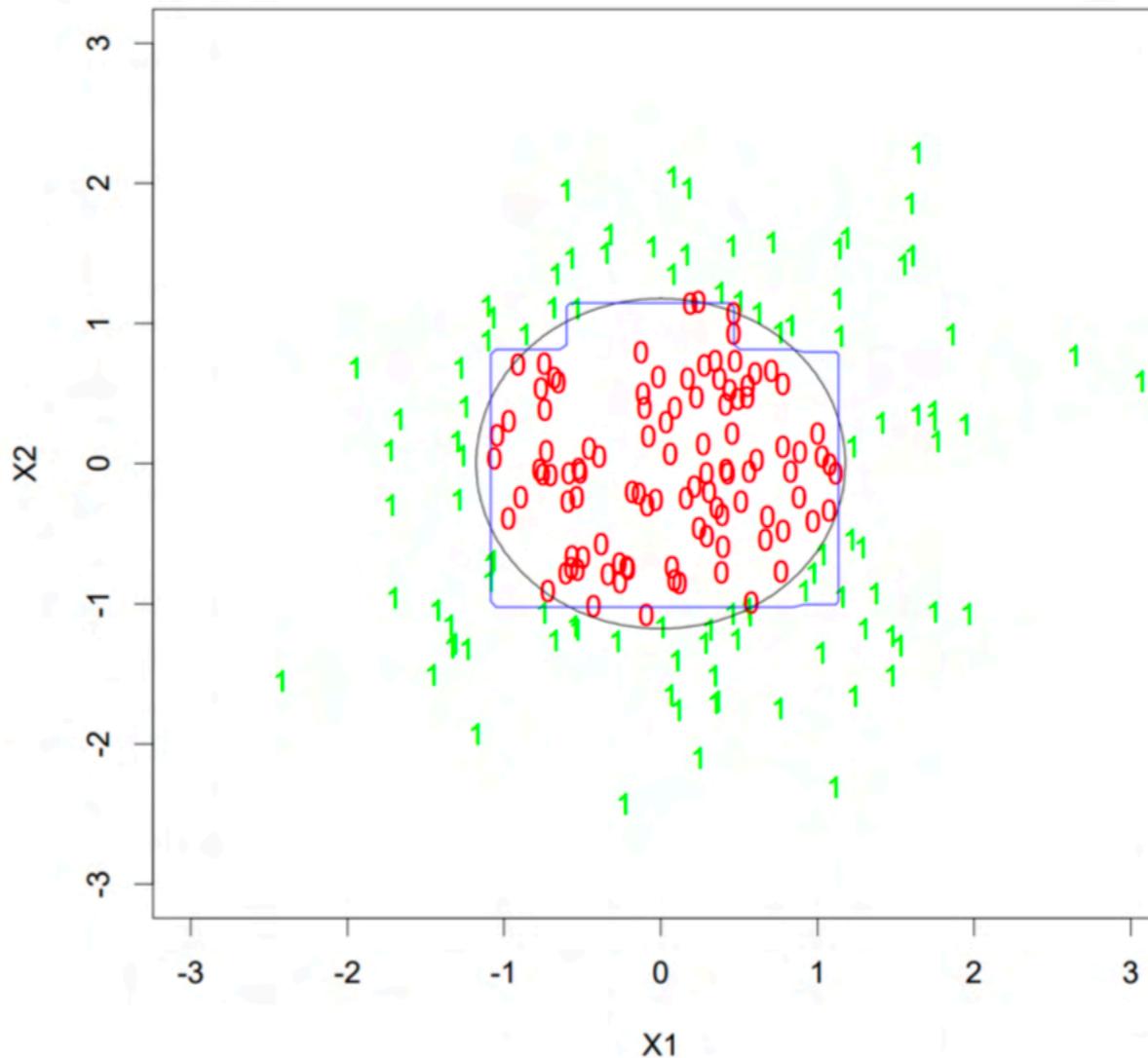
Typical decision boundaries for a single tree



Nested spheres example

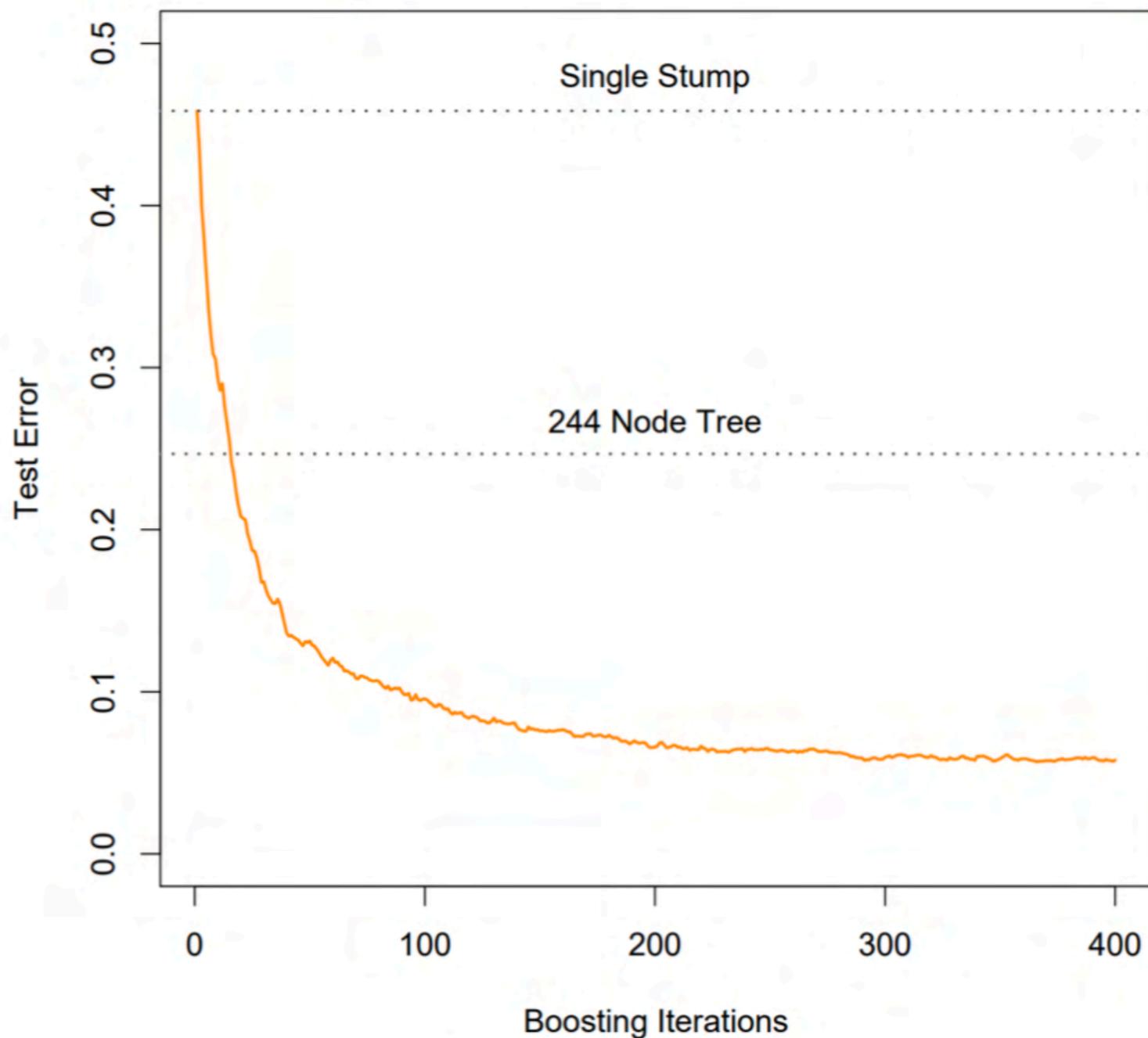
Typical decision boundaries for bagged trees

Not bad, but worse in higher dims since the boundary is refined only by chance, whereas boosting refines the boundary in a targeted way.



Nested spheres example

Boosting training results



Nested spheres example

Boosted stumps do best, perhaps because the true decision boundary has no interaction terms.

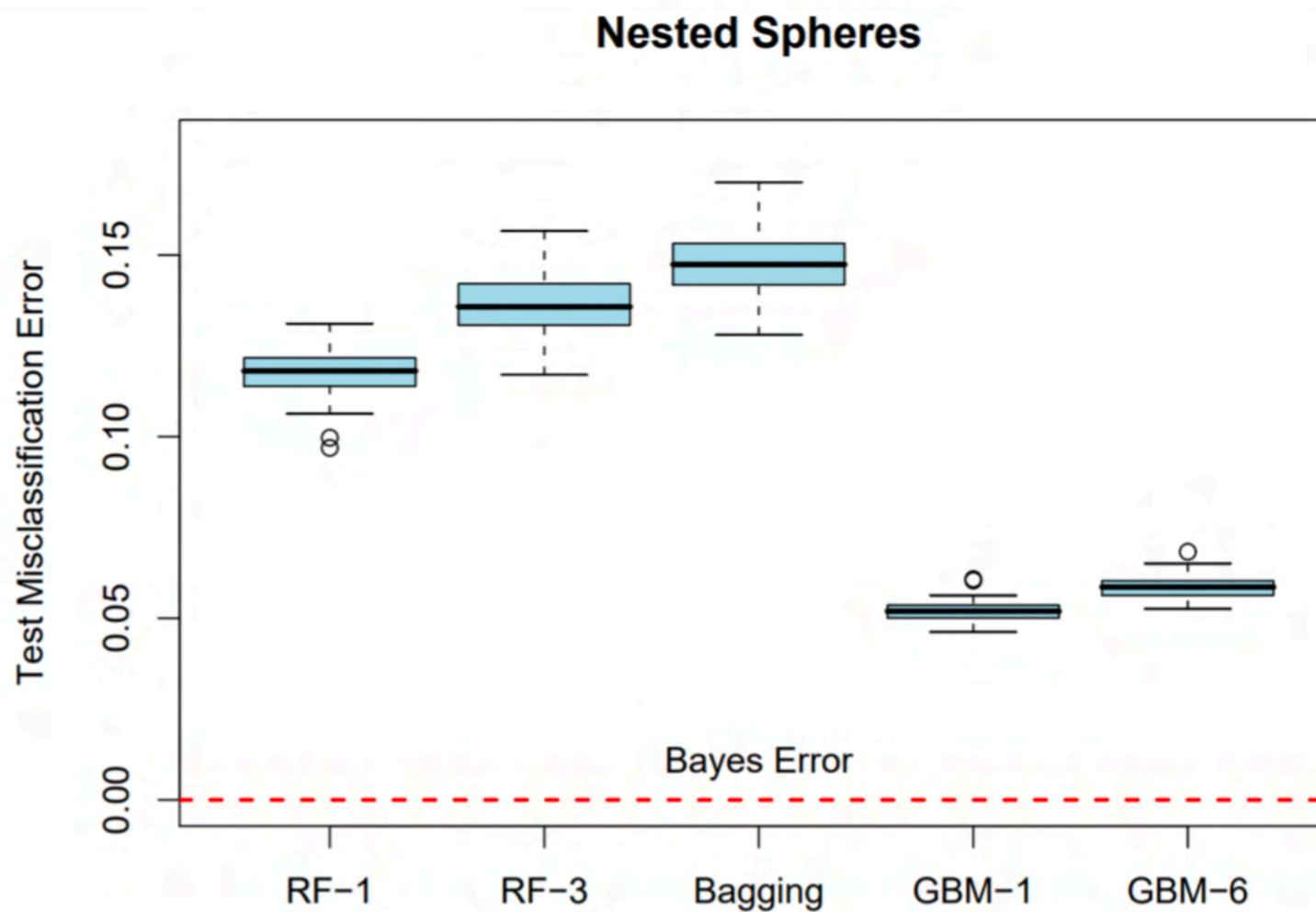


FIGURE 15.2. The results of 50 simulations from the “nested spheres” model in \mathbb{R}^{10} . The Bayes decision boundary is the surface of a sphere (additive). “RF-3” refers to a random forest with $m = 3$, and “GBM-6” a gradient boosted model with interaction order six; similarly for “RF-1” and “GBM-1.” The training sets were of size 2000, and the test sets 10,000.

Nested spheres example

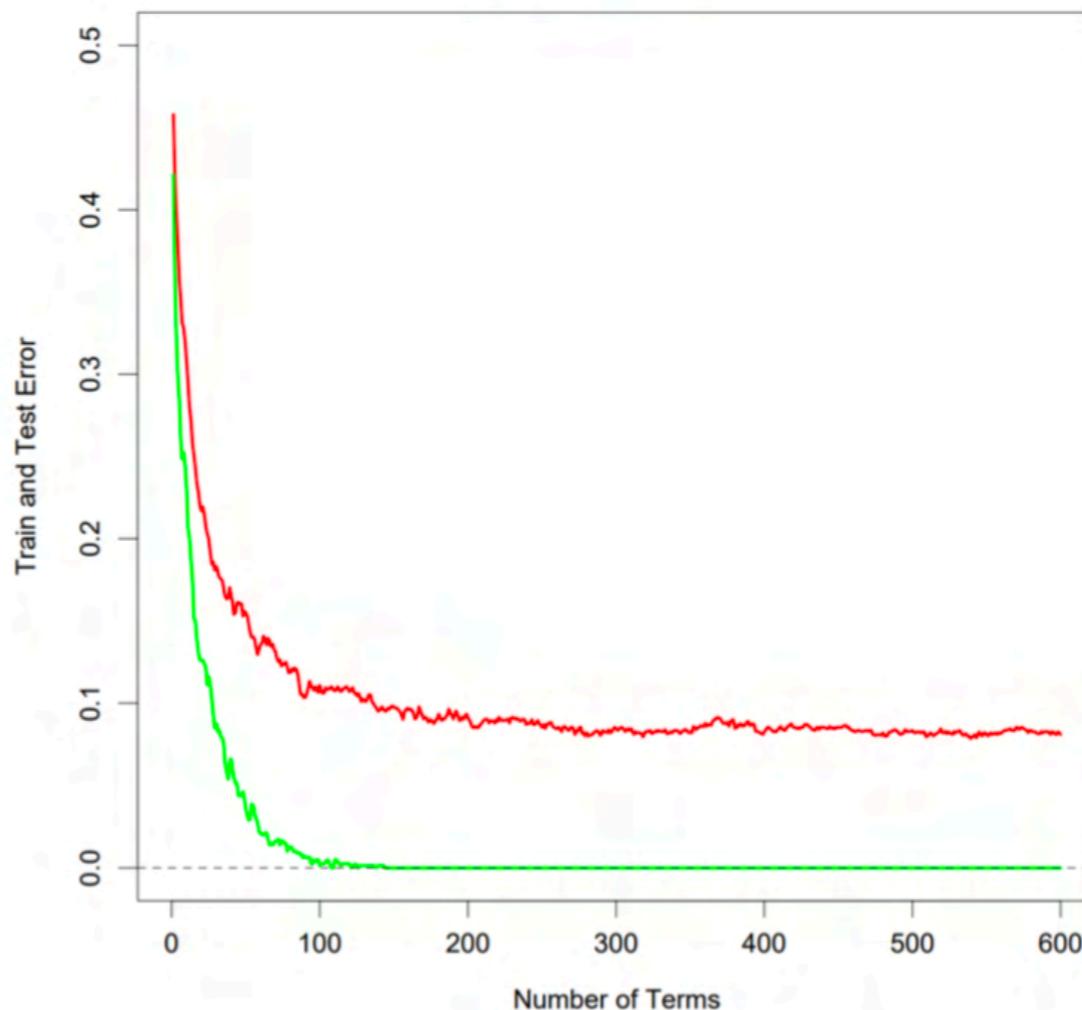
Train (green) and test (red) error as B grows.

Test error continues going down even if training error is zero!

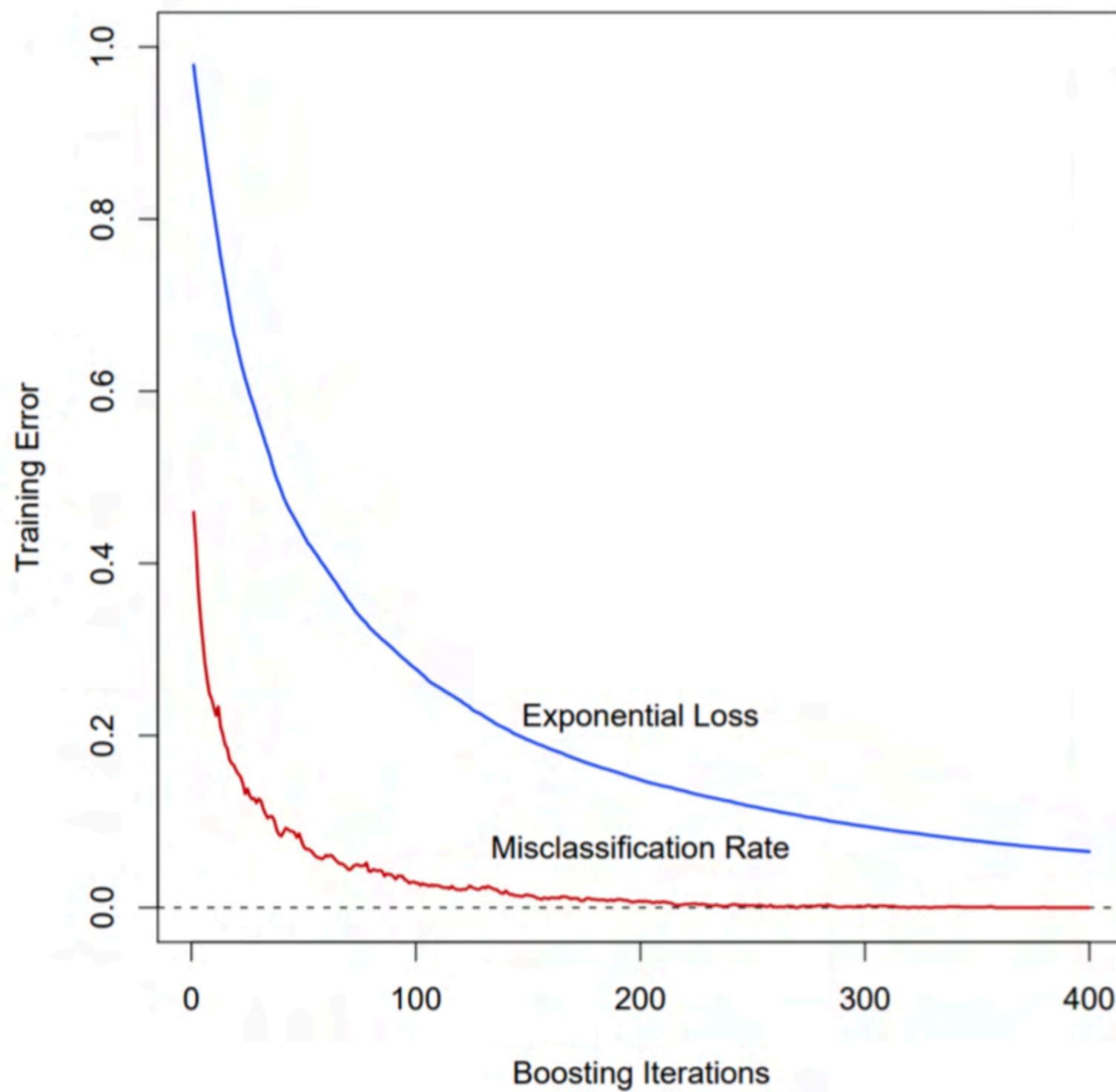
How is this possible? And why doesn't it overfit?

Nested spheres in R^{10} — Bayes error is 0%.

Stumps

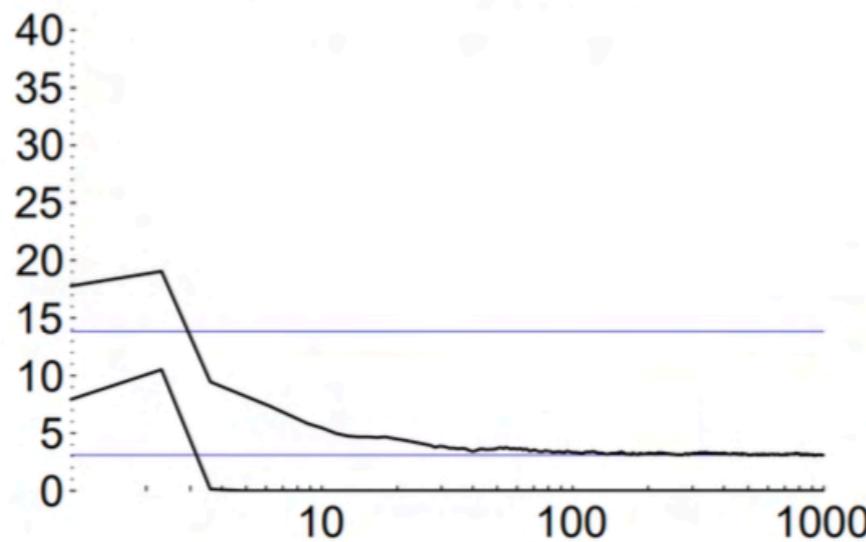


Exponential loss vs error rate

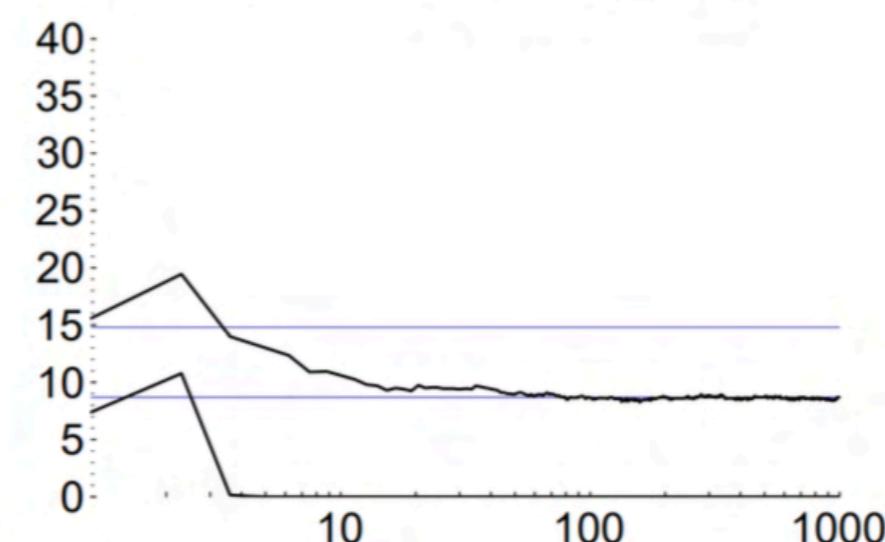


Overfitting?

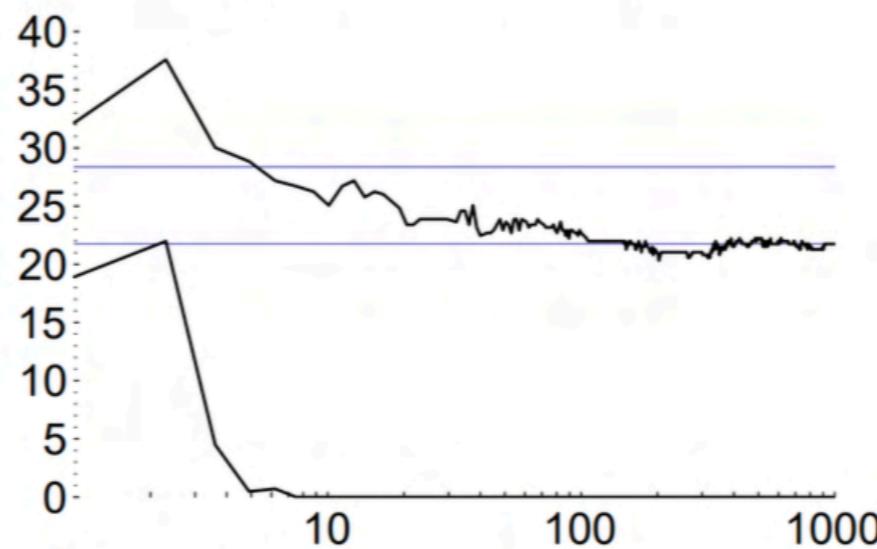
letter dataset



satimage dataset



vehicle dataset

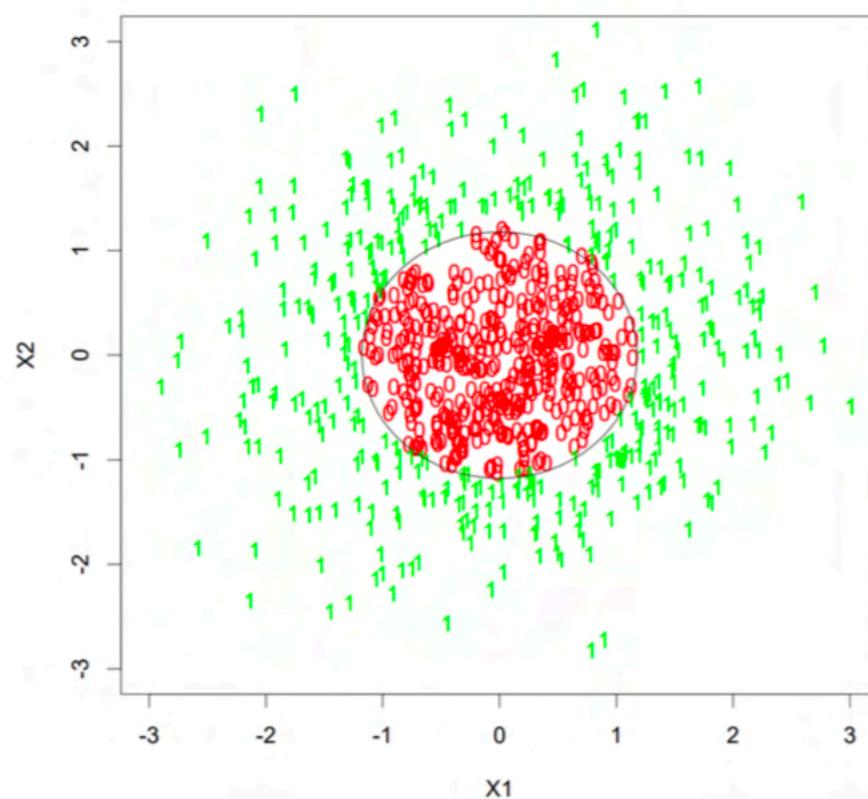


Overfitting?

- Early on, it was often observed that test error continued to decrease as the number of stages B increased.
- Consequently, it was sometimes claimed that boosting does not overfit.
- However, these experiments tended to be in cases where the classes are perfectly separable.
 - That is, cases where a true error rate of 0 can be achieved.
- Schapire et al. (1998) propose an explanation in these cases.
- In more general situations, boosting can overfit when B is too large.

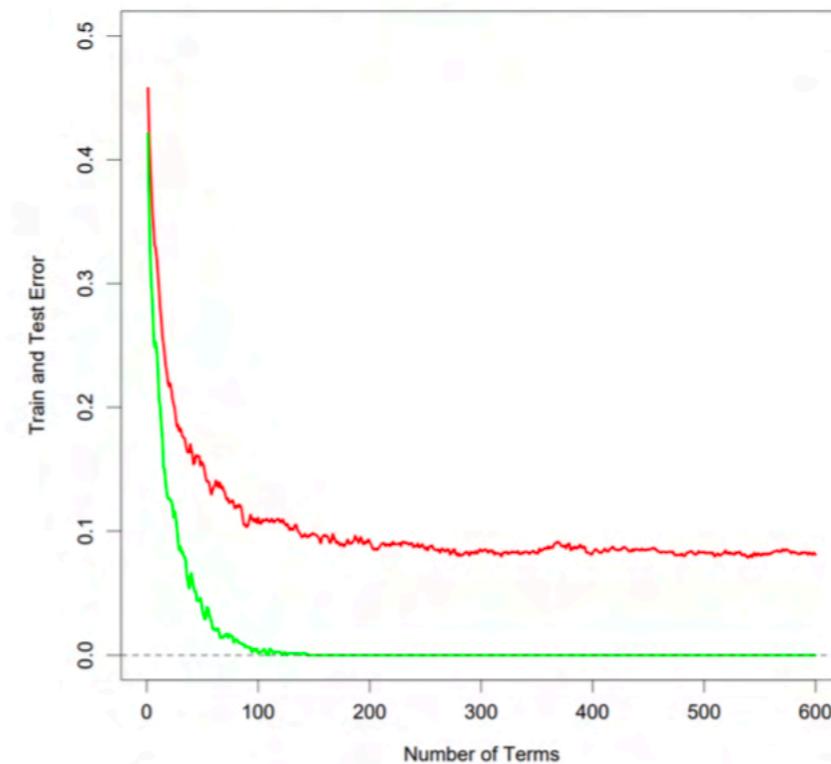
Overfitting?

If classes are perfectly separable, boosting doesn't seem to overfit.



Nested spheres in R^{10} — Bayes error is 0%.

Stumps

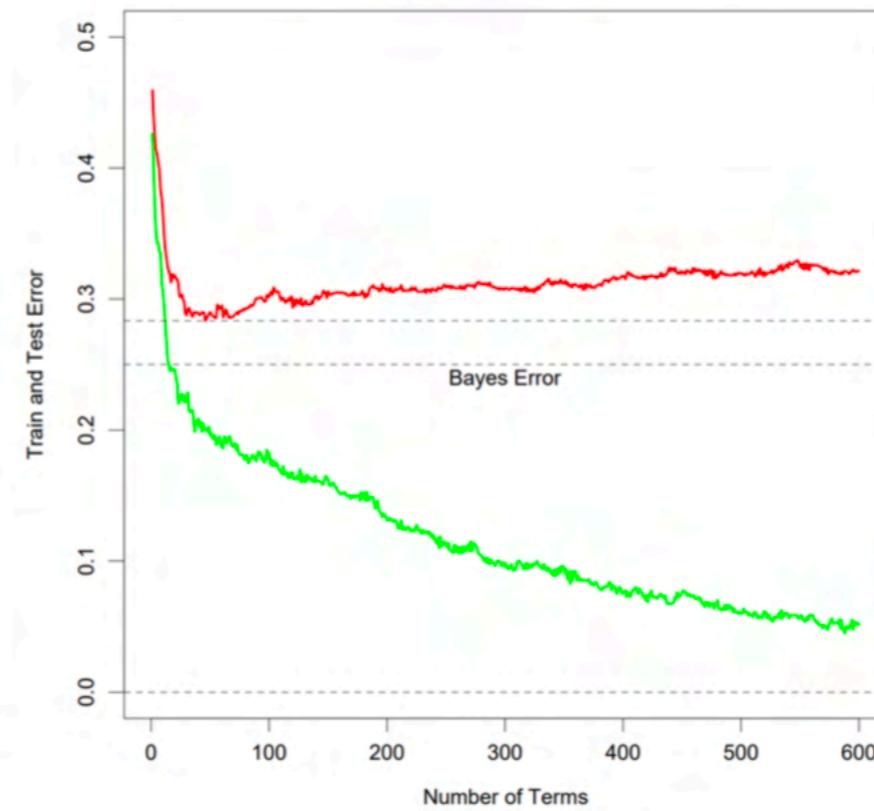
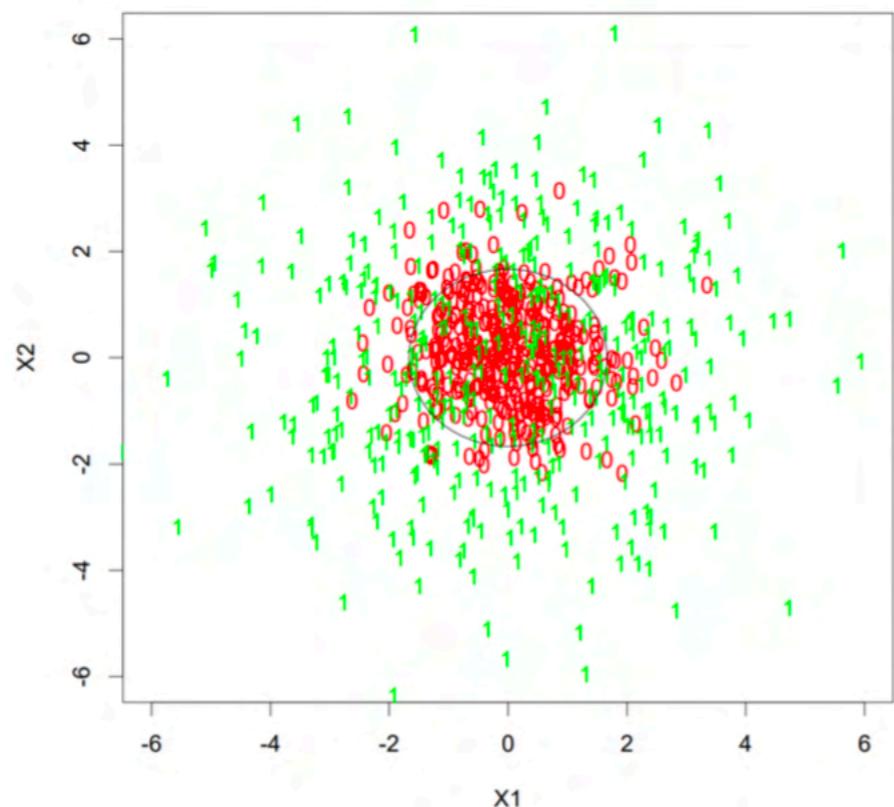


Overfitting?

If classes overlap, boosting does overfit.

Nested Gaussians in R^{10} — Bayes error is 25%.

Stumps



How to choose B

- Need to choose B (number of trained models) appropriately.
- Overfitting can occur if B is too large.
- Cross-validation can be used to choose B , as usual.
- In some cases, cross-validation may be too slow.
- An alternative is to do a single train/test split and choose a B that yields good performance on the test set.

Variations of Boosting

Stochastic boosting

Gradient boosting

XGboost

Stochastic Boosting

Choose $\lambda, \alpha \in (0, 1]$

Algorithm:

Initialize $\hat{f}(x) = 0$

For $b = 1, 2, \dots, B$

Draw a random set $S \subseteq \{1, \dots, n\}$ of size αn

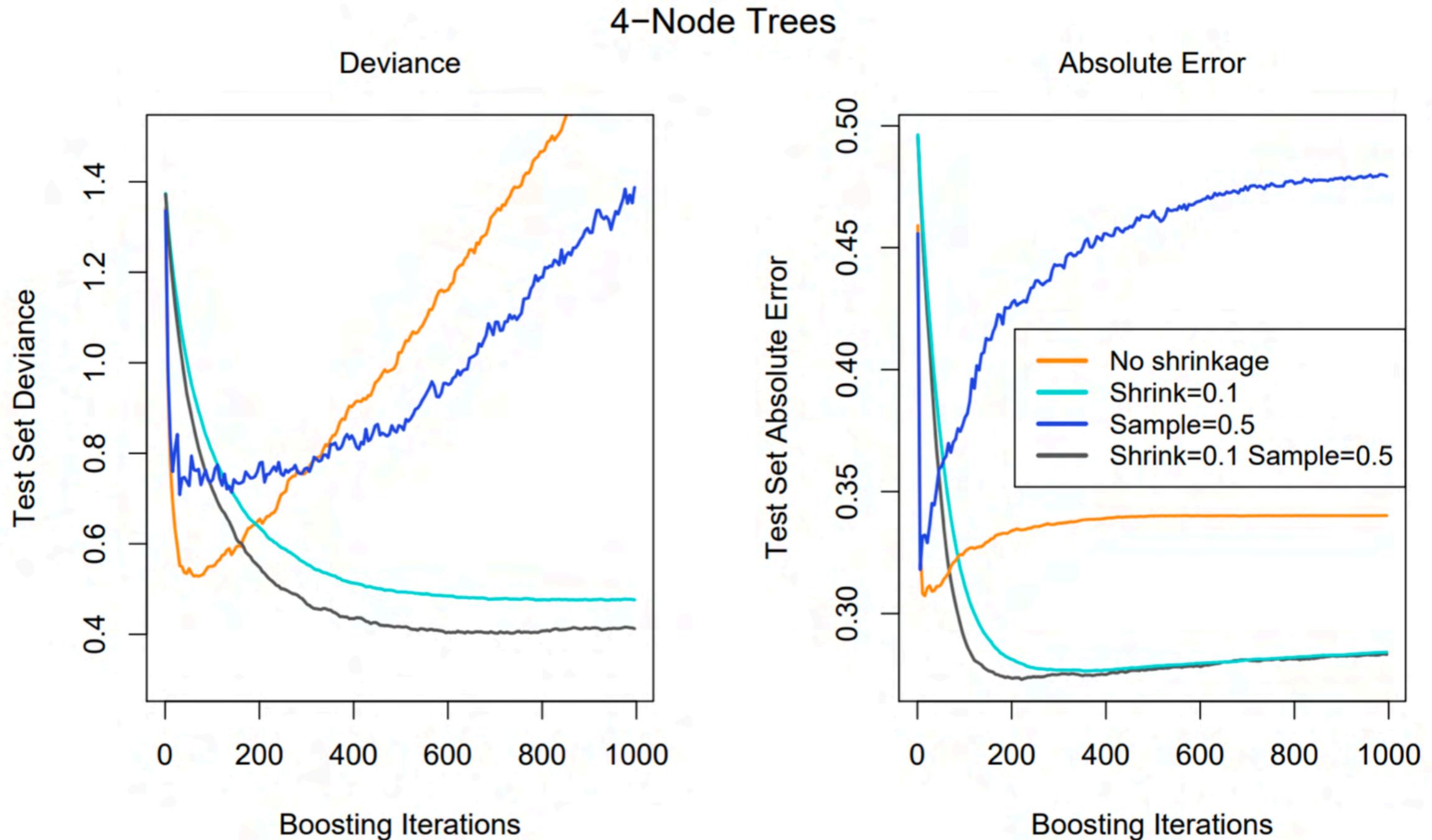
$$\hat{f}_b(x) = \underset{h_b \in \mathcal{H}}{\operatorname{argmin}} \sum_{i \in S}^n L(y_i, \hat{f}(x_i) + h_b(x_i))$$

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}_b(x)$$

Randomization can improve performance and speed.

It helps control variance, similar to bagging.

Stochastic Boosting



Gradient Boosting

Sometimes it is hard to solve $\hat{f}_b(x) = \operatorname{argmin}_{h_b \in \mathcal{H}} \sum_{i \in S}^n L(y_i, \hat{f}(x_i) + h_b(x_i))$

Recall the regression algorithm:

Algorithm:

Set $\hat{f}(x) = 0$ **and** $r_i = y_i$ **for all** $i = 1, \dots, n$

For $b = 1, 2, \dots, B$

(a) Fit $\hat{f}^b(x)$ **via a given method using the training data** $\{(r_i, x_i)\}_{i=1}^n$

(b) Update estimator \hat{f} : $\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$

(c) Update the residuals: $r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$ \leftarrow **Residuals**

Output the boosted model:

$$\hat{f}(x) = \sum_{b=1}^B \lambda \hat{f}^b(x)$$

Gradient Boosting

Idea: Gradient is pseudo-residual

Algorithm (Gradient Boosting)

$$\hat{f}(x) = \operatorname{argmin}_{\beta \in \mathbb{R}} \sum_{i \in S}^n L(y_i, \beta)$$

For $b = 1, 2, \dots, B$

- (a) **Compute pseudo-residuals:** $r_i \leftarrow -\left[\frac{\partial L(y_i, f)}{\partial f} \right]_{f=\hat{f}(x_i)}$
- (b) **Fit** $\hat{f}^b(x)$ **via a given method using the training data** $\{(r_i, x_i)\}_{i=1}^n$
- (c) **Find optimal shrinkage** $\lambda \leftarrow \operatorname{argmin}_{\lambda \in (0,1]} \sum_{i=1}^n L(y_i, \hat{f}(x_i) + \lambda \hat{f}_b(x))$
- (d) **Update** $\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}_b(x)$

If we use CART in Step (b), the algorithm is called **gradient boosting trees**

Why XGBoost is so popular?

Breiman said in 1996 that boosted trees were the “best off-the-shelf classifier in the world”.

In the last 5–10 years, boosted trees have emerged as a clear leader in many ML competitions.

Recent Kaggle winning solutions (Non-image tasks)

- Mercedes-Benz Auto Testing Time — XGBoost
- Instacart Market Basket Analysis — XGBoost
- March Machine Learning Mania (2017 NCAA tournament) — Logistic regression
- Two Sigma Financial Modeling — ensemble of Extra-Trees and Ridge regression
- Outbrain Click Prediction — XGBoost+NN ensemble of around 50 models
- Allstate Claims Severity — ensemble of many models
- Santander Product Recommendation — ensembled XGBoost
- Seizure Prediction — ensemble of XGBoost, KNN, SVM, Logistic regression
- Bosch Production Line Performance — XGBoost
- Red Hat Business Value — XGBoost

Graphical Model

Junwei Lu



HARVARD
T.H. CHAN

SCHOOL OF PUBLIC HEALTH
Department of Biostatistics



Undirected Graphical Model

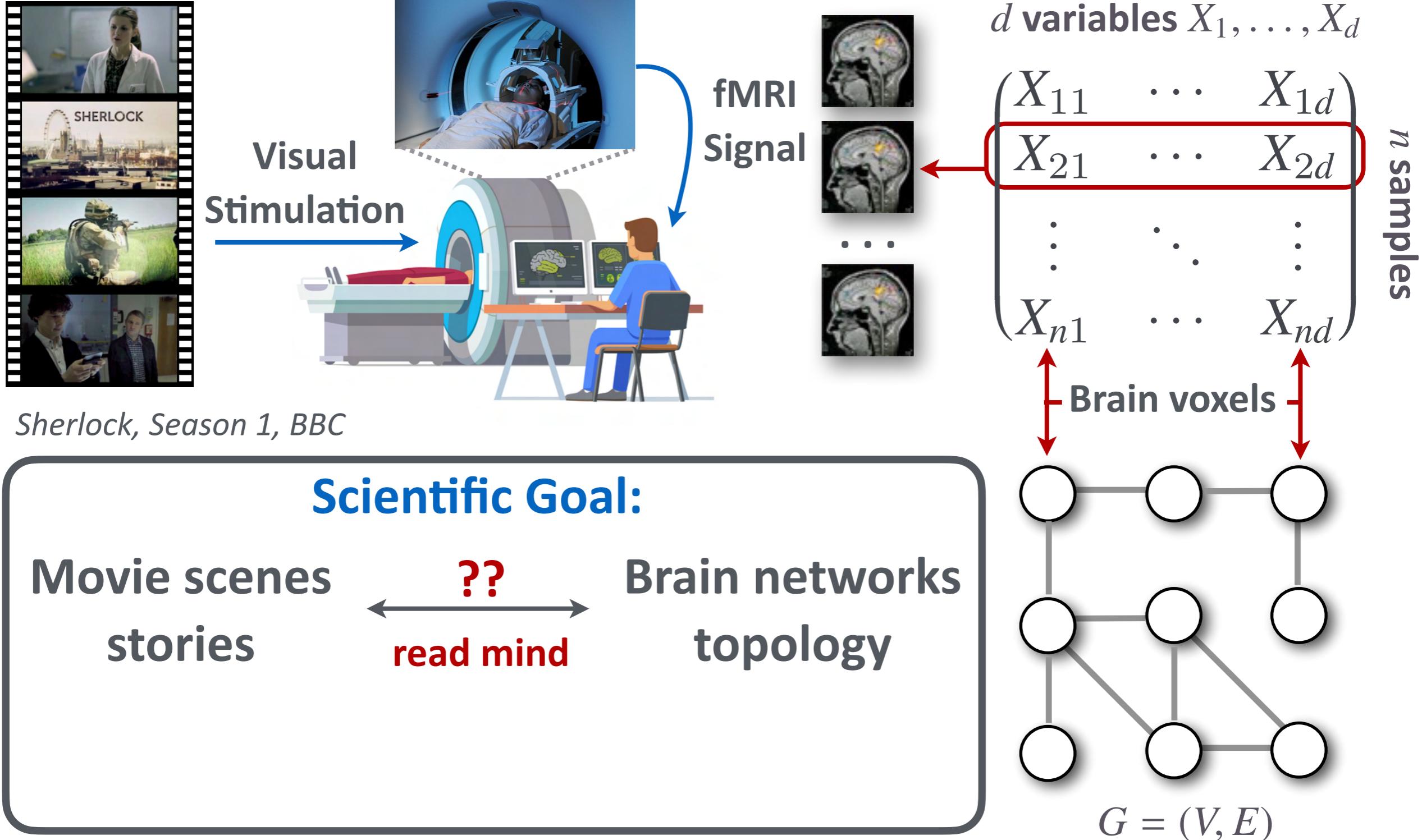
Graphical model provides network representation of dependency structures complex dataset

- **Genomic network**
- **Brain network**
- **Financial stock network**

Correlation can be propagated via variables:

Corr(Ice Cream Sale, Shark Attack) is high because of the intermediate variable “temperature”

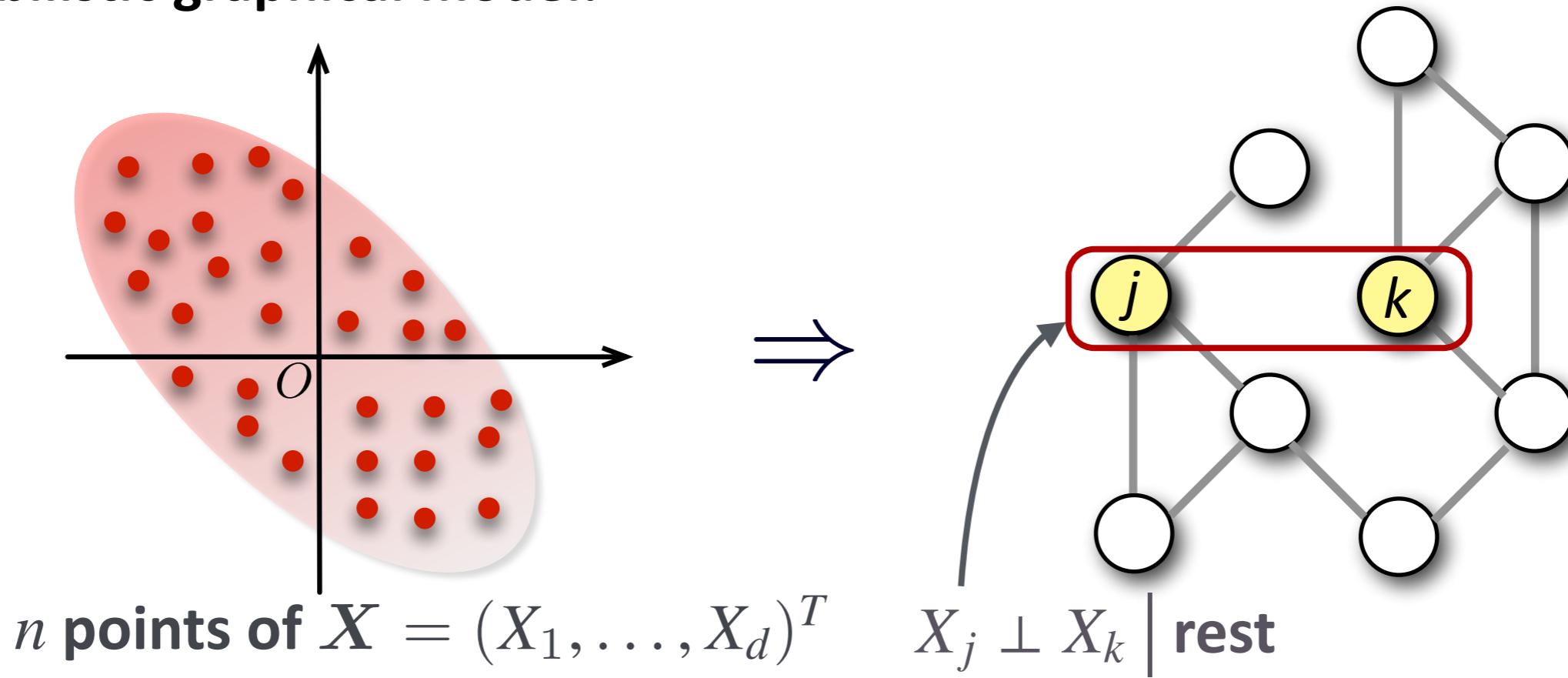
Applications to fMRI data



Undirected Graphical Model

Idea: Two variables are not connected if they are **conditionally independent**

Probabilistic graphical model:



The nodes j and k are disconnected, if $X_j \perp X_k | X_{\setminus\{j,k\}}$

where $X_{\setminus A} := \{X_j, j \notin A\}$

or $p(X_j, X_k | X_{\setminus\{j,k\}}) = p(X_j | X_{\setminus\{j,k\}})p(X_k | X_{\setminus\{j,k\}})$

Gaussian Graphical Model

If the data is Gaussian: $X \sim N(\mu, \Sigma)$

Key: The precision matrix $\Theta = \Sigma^{-1}$

$\Theta_{jk} = 0$ if and only if $X_j \perp X_k | X_{\setminus\{j,k\}}$

The sparsity of precision matrix encodes the conditional dependency structure of Gaussian graphical model

However, when $d > n$, the sample covariance matrix $\hat{\Sigma}$ is not invertible

GLasso Estimator

GLasso estimates the precision matrix via MLE + L1 penalty

$$\min_{\Theta} \underbrace{\text{tr}(\widehat{\Sigma}\Theta) - \log |\Theta|}_{\text{Negative Gaussian log-likelihood}} + \lambda \underbrace{\sum_{j,k} |\Theta_{jk}|}_{\ell_1\text{-regularization}}$$

We have L1-penalty because we believe the graph is sparse.

Ising Model

The density of Gaussian likelihood: x is continuous

$$p(x) \propto \exp\left(-\frac{1}{2}x^T \Theta x\right) = \exp\left(-\frac{1}{2} \sum_{j,k} \Theta_{jk} x_j x_k\right)$$

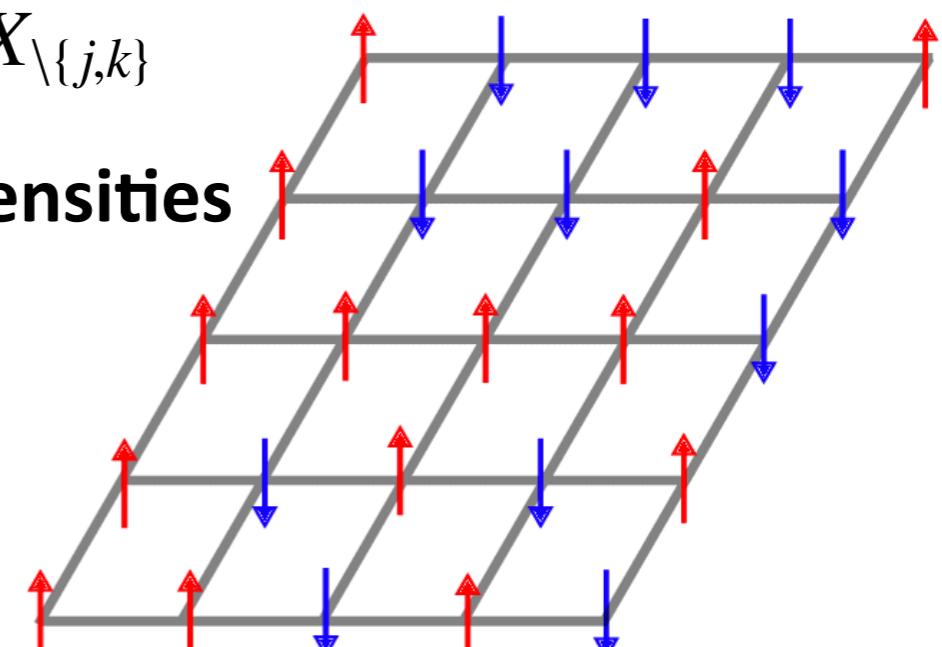
$\Theta_{jk} = 0$ if and only if $X_j \perp X_k | X_{\setminus\{j,k\}}$

Ising model: $x \in \{-1, 1\}^d$ is binary

$$\mathbb{P}(X = x) \propto \exp\left(-x^T \Theta x\right) = \exp\left(\sum_{j,k} \Theta_{jk} x_j x_k\right)$$

We also have $\Theta_{jk} = 0$ if and only if $X_j \perp X_k | X_{\setminus\{j,k\}}$

The only difference is the normalizer of two densities



Ising Model

Ising model: $x \in \{-1, 1\}^d$ is binary

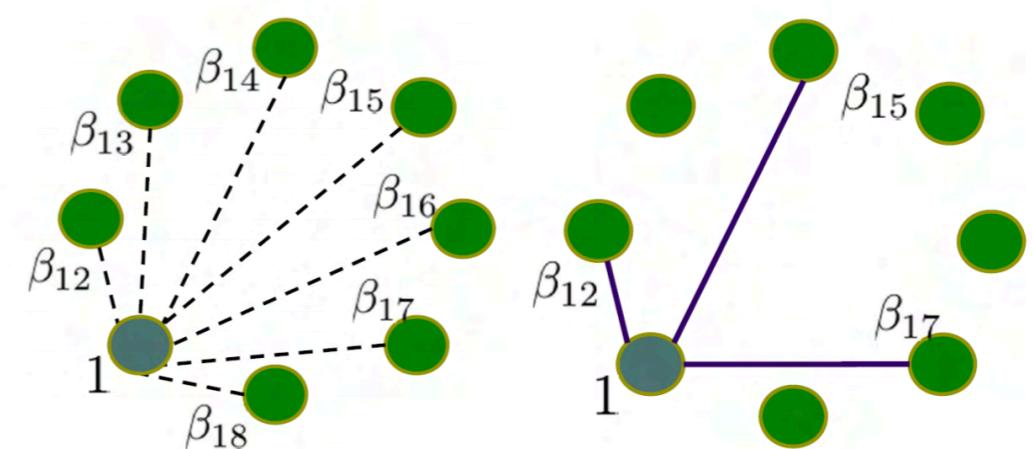
$$\mathbb{P}(X = x) \propto \exp\left(-x^T \Theta x\right) = \exp\left(\sum_{j,k} \Theta_{jk} x_j x_k\right)$$

We estimate the Ising graph using the property:

$$\mathbb{P}(X_j | X_{\setminus j}) = \text{logistic}\left(2X_j \sum_{k \neq j} \Theta_{jk} X_k\right), \text{ where } \text{logistic}(x) = 1/(1 + e^{-x})$$

Nodewise regression for Ising model

$$\hat{\beta} = \operatorname{argmin}_{\beta} \sum_{i=1}^n \text{logistic}(2X_{ij}\beta^T X_{i\setminus j}) + \lambda \|\beta\|$$



Reinforcement Learning

Junwei Lu



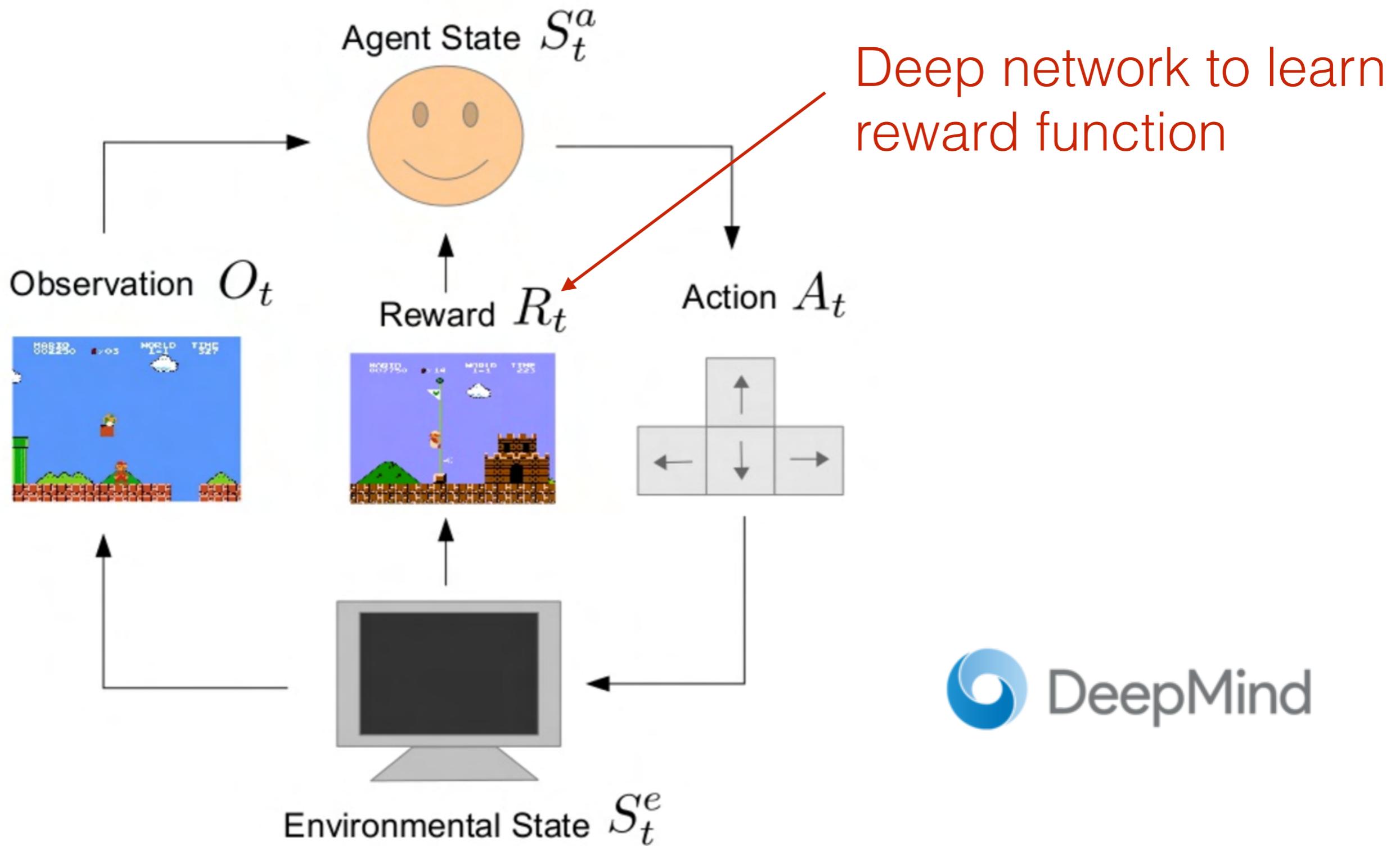
HARVARD
T.H. CHAN

SCHOOL OF PUBLIC HEALTH
Department of Biostatistics



Reinforcement Learning

Reinforcement learning

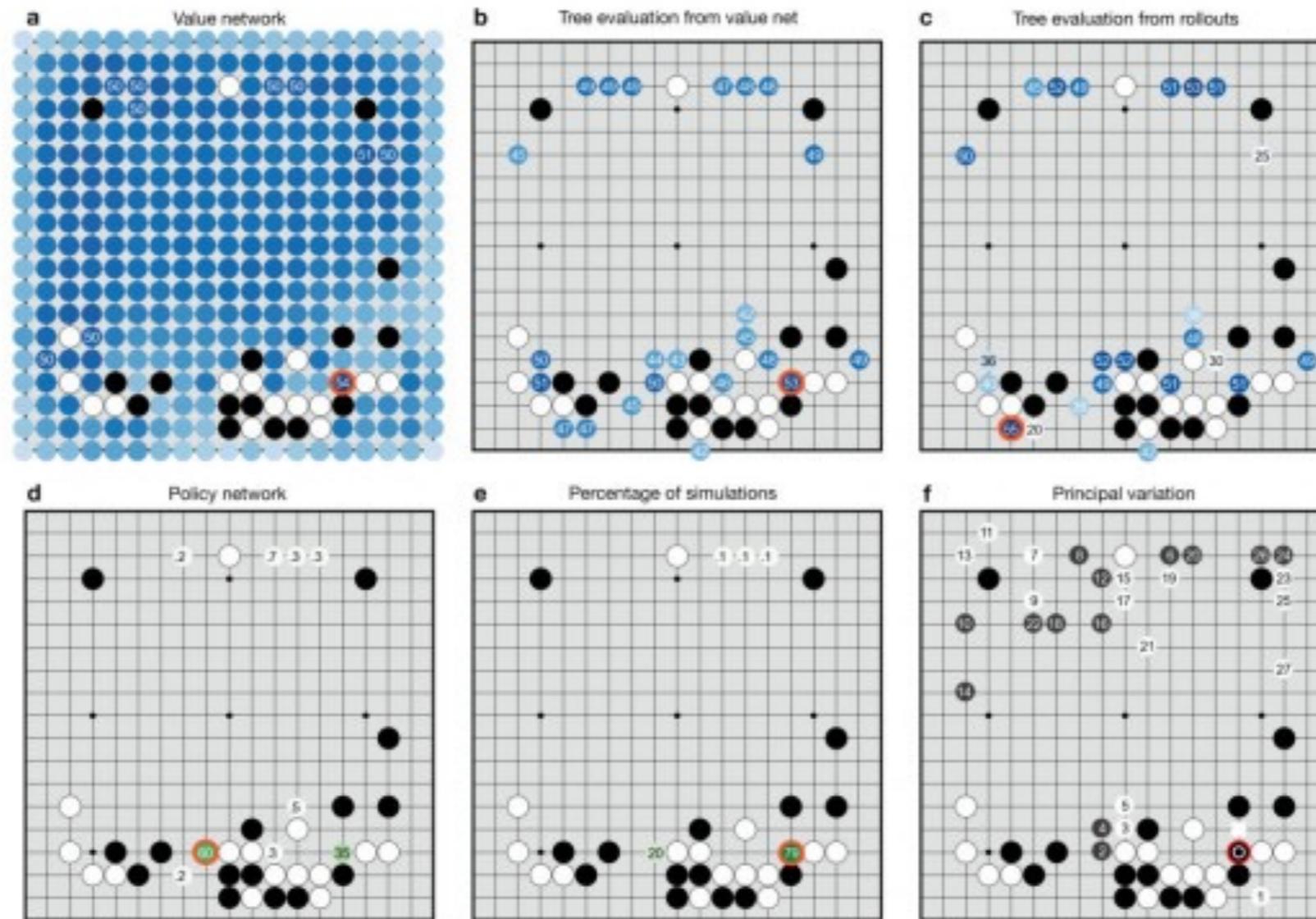


Reinforcement Learning Applications

Reinforcement learning



DeepMind



AlphaGo

Reinforcement Learning Applications

Reinforcement learning

CMU ARTIFICIAL INTELLIGENCE IS TOUGH POKER PLAYER

Libratus builds substantial lead in Brains vs. AI competition

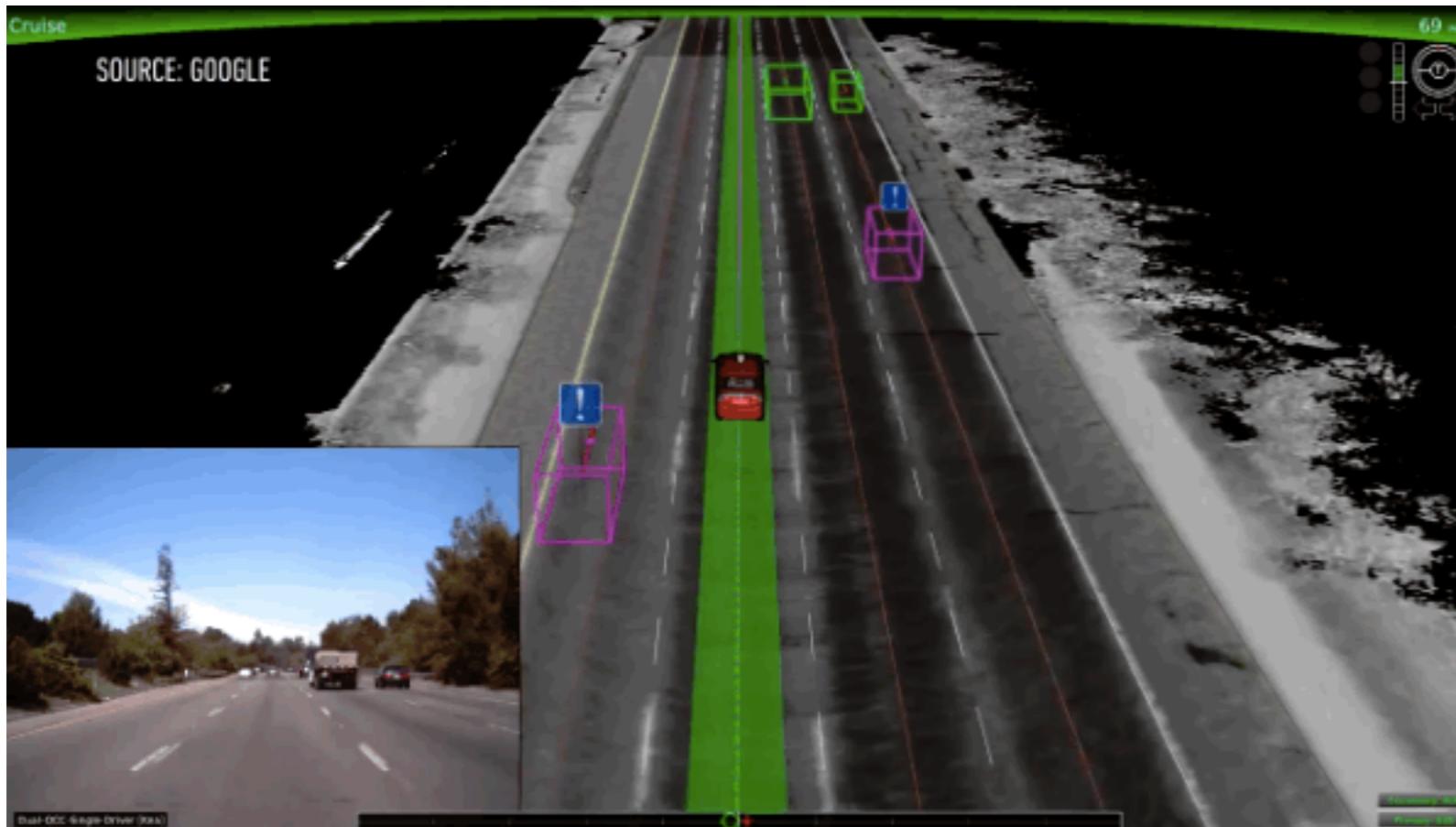
By [Byron Spice](#) (CMU) and [Garrett Allen](#) (Rivers Casino)



Reinforcement Learning Applications

Reinforcement learning + image

Self-driving



Reinforcement Learning in Public Health

Mobile Health



Clinical Trials / Treatment strategy

Reinforcement Learning Applications

Clinical Trials / Treatment strategy

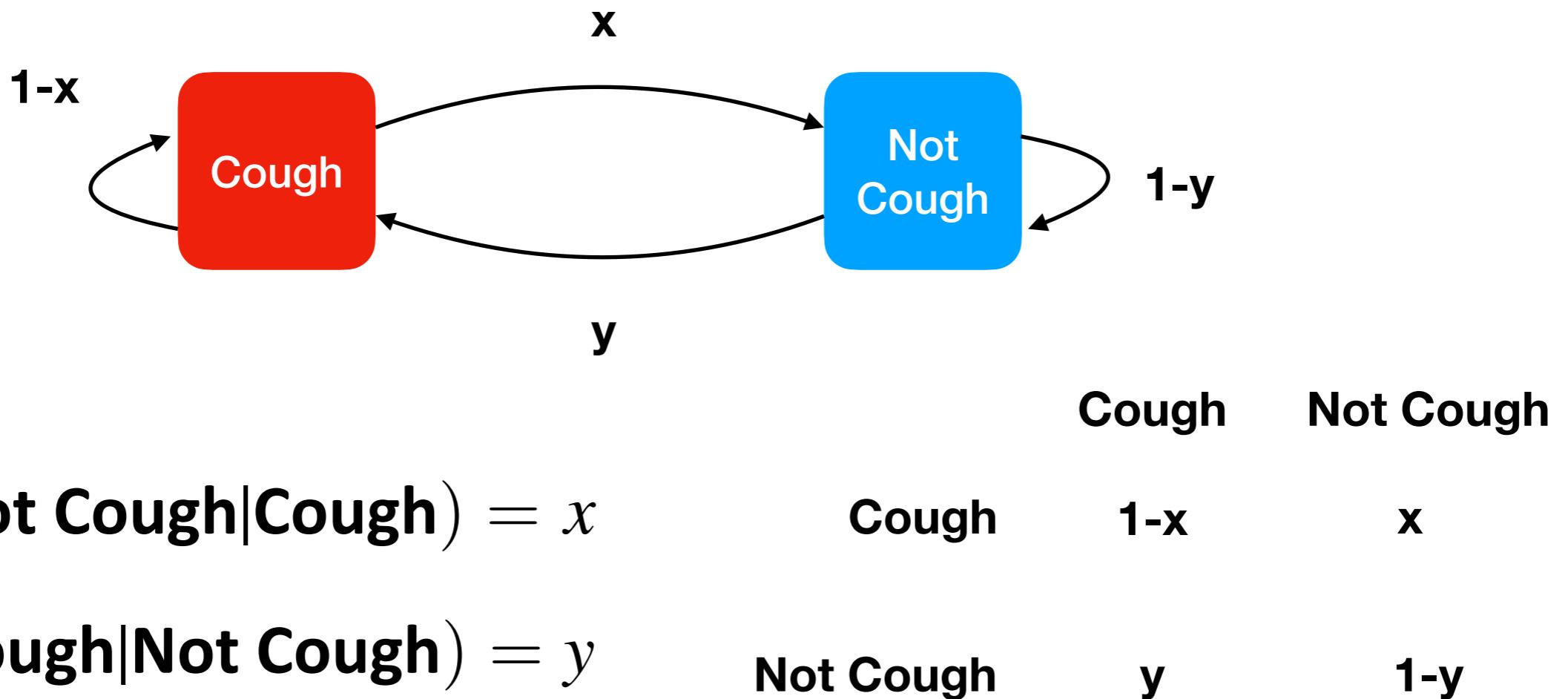
State: Cough / Not Cough

Action: Take medicine / Not take medicine

Reward: Body temperature

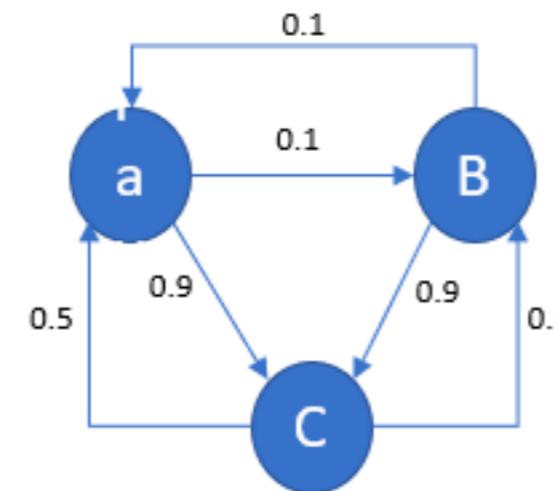
Markov Chain

Markov chain is a system which is in a certain state at each step, with the state changing randomly between steps



Markov Chain

Transition probability



$$P = \begin{bmatrix} 0 & 0.1 & 0.9 \\ 0.1 & 0.0 & 0.9 \\ 0.5 & 0.5 & 0.0 \end{bmatrix}$$

P_s is the state transition probabilities

$P_s(s')$ = the transition probability from s to s'

Markov decision processes

A Markov decision process is a tuple $(S, A, \{P_{sa}\}, \gamma, R)$, where:

S is a set of state, A is a set of actions

- P_{sa} are the state transition probabilities. For each state $s \in S$ and action $a \in A$, P_{sa} is a distribution over the state space. We'll say more about this later, but briefly, P_{sa} gives the distribution over what states we will transition to if we take action a in state s .
- $\gamma \in [0, 1)$ is called the **discount factor**.
- $R : S \times A \mapsto \mathbb{R}$ is the **reward function**. (Rewards are sometimes also written as a function of a state S only, in which case we would have $R : S \mapsto \mathbb{R}$).

State: Cough / Not Cough

Action: Take medicine / Not take medicine

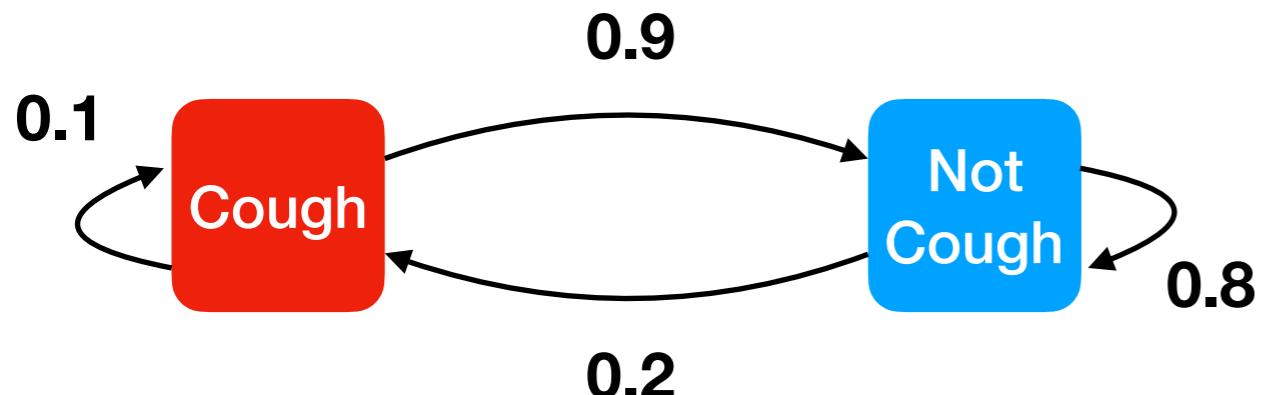
Reward: Body temperature

Markov decision processes

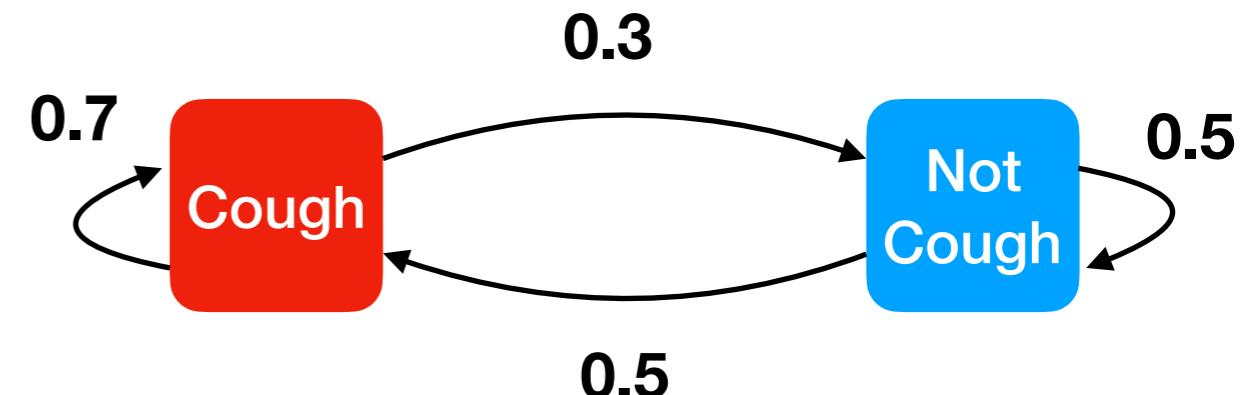
A Markov decision process is a tuple $(S, A, \{P_{sa}\}, R)$, where:

S is a set of states, A is a set of actions

- P_{sa} are the state transition probabilities. For each state $s \in S$ and action $a \in A$, P_{sa} is a distribution over the state space. We'll say more about this later, but briefly, P_{sa} gives the distribution over what states we will transition to if we take action a in state s .
- $\gamma \in [0, 1)$ is called the **discount factor**.
- $R : S \times A \mapsto \mathbb{R}$ is the **reward function**. (Rewards are sometimes also written as a function of a state S only, in which case we would have $R : S \mapsto \mathbb{R}$).



$a = \text{Take medicine}$



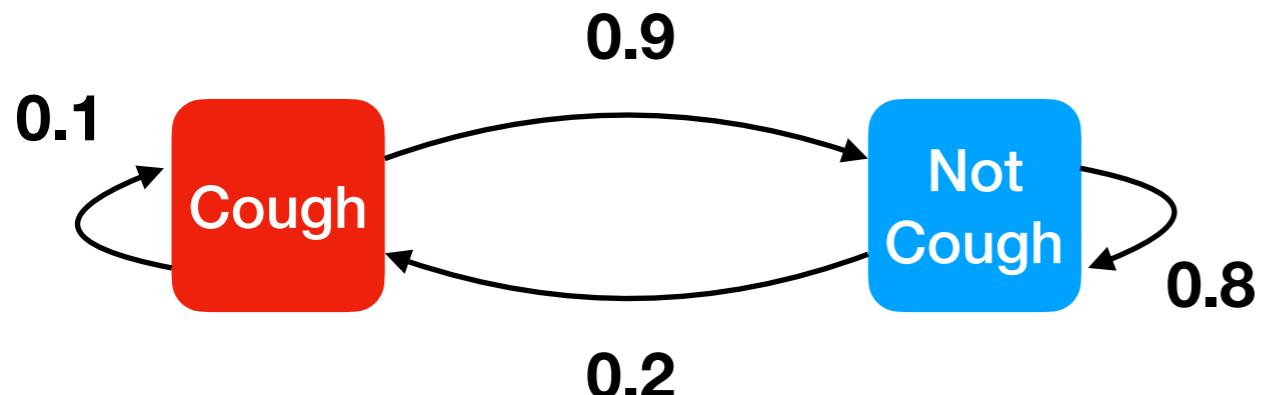
$a = \text{Not take medicine}$

Markov decision processes

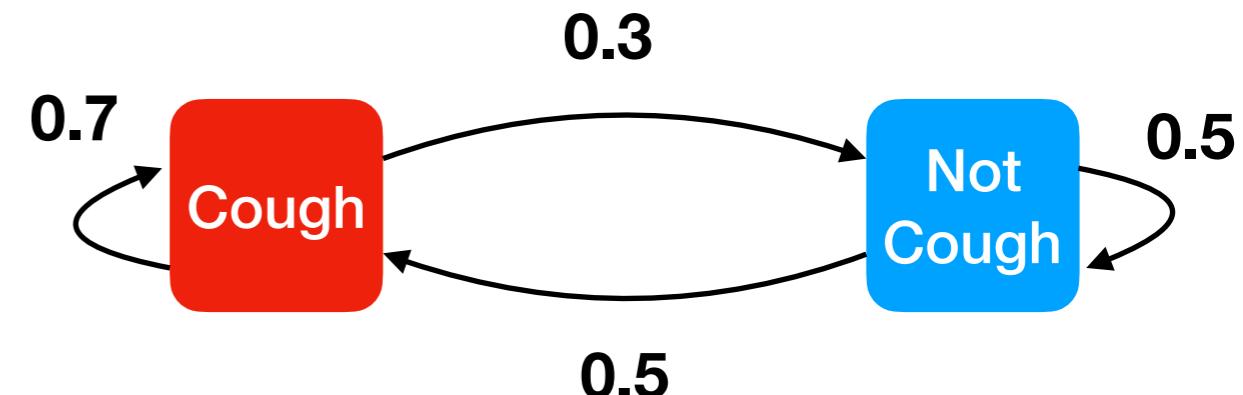
A Markov decision process is a tuple $(S, A, \{P_{sa}\}, R)$, where:

S is a set of states, A is a set of actions

- P_{sa} are the state transition probabilities. For each state $s \in S$ and action $a \in A$, P_{sa} is a distribution over the state space. We'll say more about this later, but briefly, P_{sa} gives the distribution over what states we will transition to if we take action a in state s .
- $\gamma \in [0, 1)$ is called the **discount factor**.
- $R : S \times A \mapsto \mathbb{R}$ is the **reward function**. (Rewards are sometimes also written as a function of a state S only, in which case we would have $R : S \mapsto \mathbb{R}$).



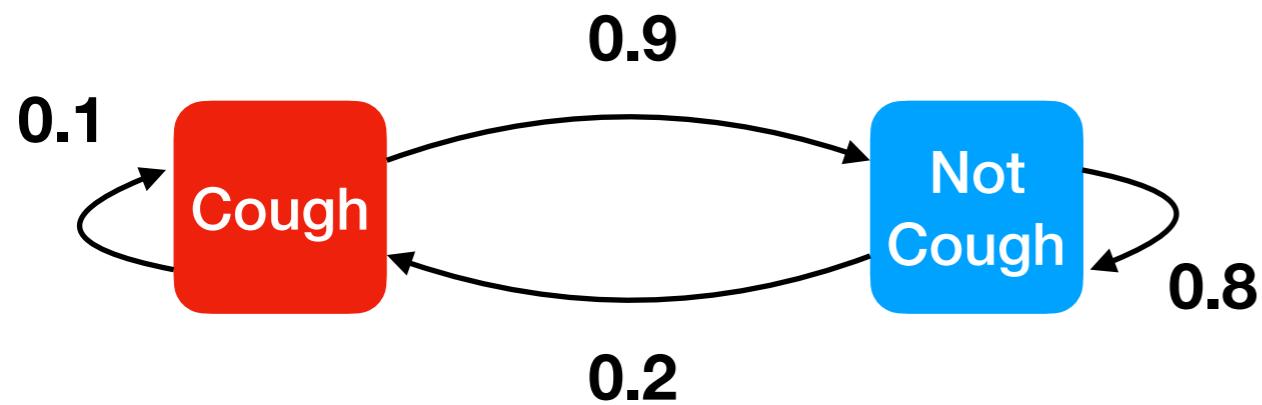
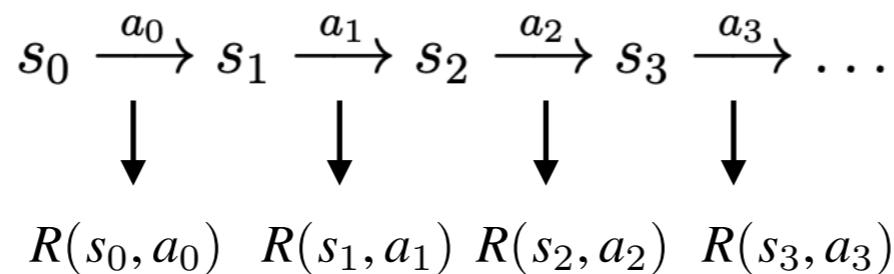
$a = \text{Take medicine}$



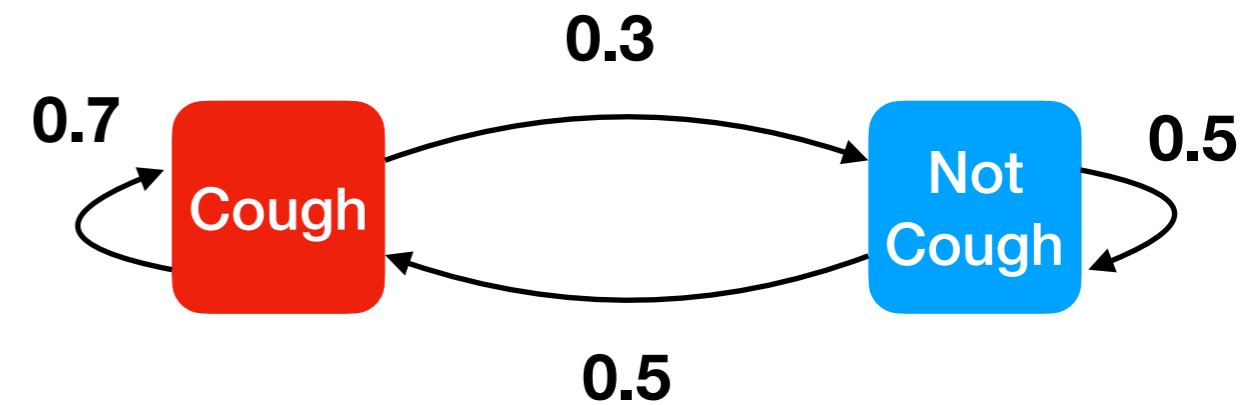
$a = \text{Not take medicine}$

Markov decision processes

The dynamics of an MDP proceeds as follows: We start in some state s_0 , and get to choose some action $a_0 \in A$ to take in the MDP. As a result of our choice, the state of the MDP randomly transitions to some successor state s_1 , drawn according to $s_1 \sim P_{s_0 a_0}$. Then, we get to pick another action a_1 . As a result of this action, the state transitions again, now to some $s_2 \sim P_{s_1 a_1}$. We then pick a_2 , and so on.... Pictorially, we can represent this process as follows:

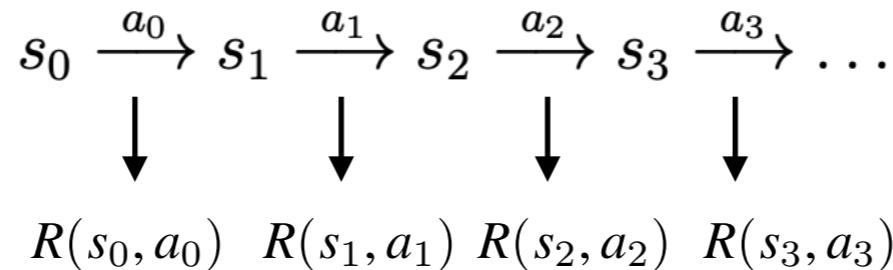


a = Take medicine



a = Not take medicine

Markov decision processes



Upon visiting the sequence of states s_0, s_1, \dots with actions a_0, a_1, \dots , our total payoff is given by

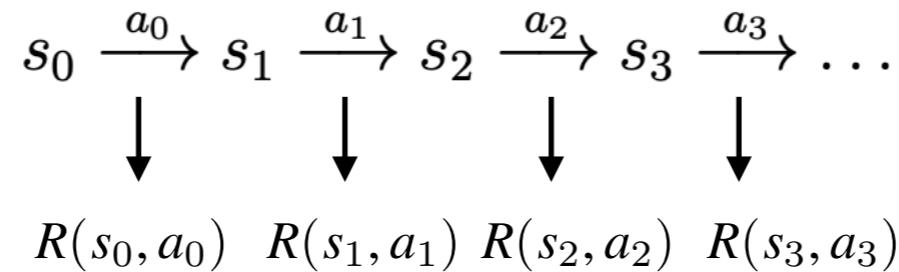
$$R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \dots .$$

Or, when we are writing rewards as a function of the states only, this becomes

$$R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots .$$

For most of our development, we will use the simpler state-rewards $R(s)$, though the generalization to state-action rewards $R(s, a)$ offers no special difficulties.

Markov decision processes



Upon visiting the sequence of states s_0, s_1, \dots with actions a_0, a_1, \dots , our total payoff is given by

$$R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \dots .$$

Or, when we are writing rewards as a function of the states only, this becomes

$$R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots .$$

For most of our development, we will use the simpler state-rewards $R(s)$, though the generalization to state-action rewards $R(s, a)$ offers no special difficulties.

Goal in reinforcement learning is to choose actions over time so as to maximize the expected value of the total payoff:

$$\mathbb{E} [R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots]$$

Markov decision processes

Goal in reinforcement learning is to choose actions over time so as to maximize the expected value of the total payoff:

$$E [R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots]$$

Note that the reward at timestep t is discounted by a factor. Thus, to make this expectation large, we would like to accrue positive rewards as soon as possible (and postpone negative rewards as long as possible).

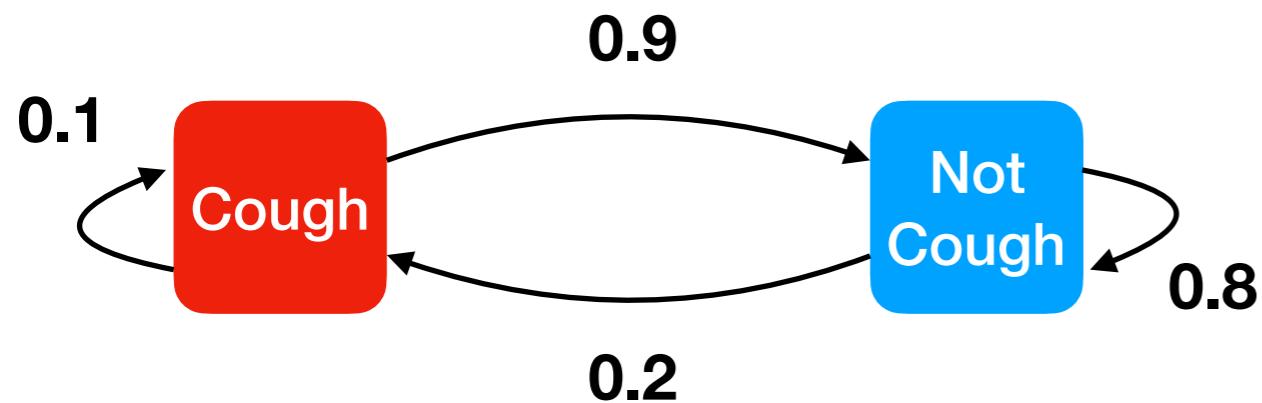
Policy Function

A **policy** is any function $\pi : S \mapsto A$ mapping from the states to the actions. We say that we are **executing** some policy π if, whenever we are in state s , we take action $a = \pi(s)$. We also define the **value function** for a policy π according to

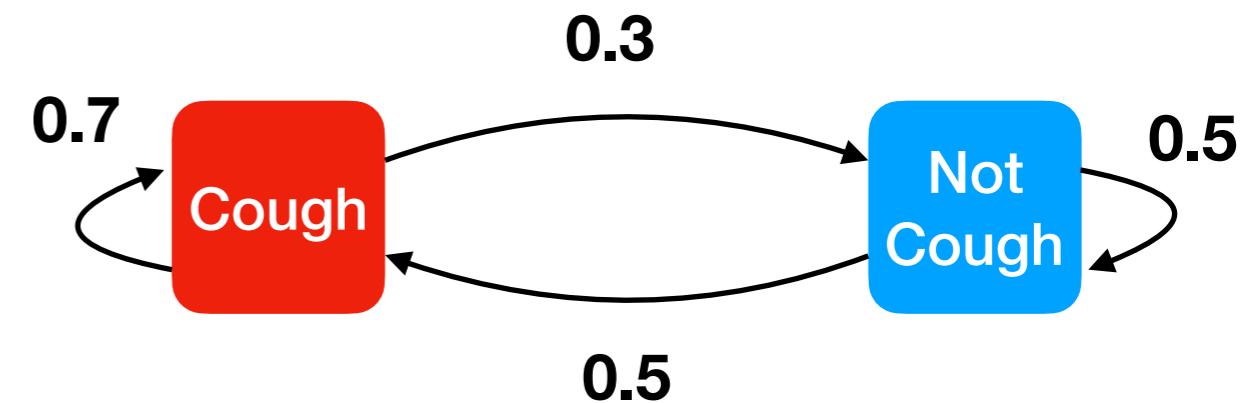
$$V^\pi(s) = \mathbb{E} [R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots | s_0 = s, \pi].$$

$V^\pi(s)$ is simply the expected sum of discounted rewards upon starting in state s , and taking actions according to π .¹

What is policy under this example?



$a = \text{Take medicine}$



$a = \text{Not take medicine}$

Bellman Equation

$$\begin{aligned} V^\pi(s) &= E[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots | s_0 = s] \\ &= E[R(s_0) + \gamma(R(s_1) + \gamma R(s_2) + \dots) | s_0 = s] \\ &= E[R(s_0) + \gamma V^\pi(s_1) | s_0 = s] \end{aligned}$$

The second term is the expected sum of discounted rewards for starting in state s'

$$E_{s' \sim P_{s\pi(s)}} [V^\pi(s')]$$

Given a fixed policy π , its value function V^π satisfies the **Bellman equations**:

$$V^\pi(s) = R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^\pi(s').$$

Bellman's equations can be used to efficiently solve for V^π

Bellman Equation

Given a fixed policy π , its value function V^π satisfies the **Bellman equations**:

$$V^\pi(s) = R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^\pi(s').$$

We have a linear equation

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} \mathcal{R}_1 \\ \vdots \\ \mathcal{R}_n \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{11} & \dots & \mathcal{P}_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$

So we can solve the value function

$$\mathbf{v} = \mathcal{R} + \gamma \mathbf{P} \mathbf{v}$$

$$\mathbf{v} = (I - \gamma \mathbf{P})^{-1} \mathcal{R}$$

Bellman Equation

We also define the **optimal value function** according to

$$V^*(s) = \max_{\pi} V^{\pi}(s). \quad (1)$$

In other words, this is the best possible expected sum of discounted rewards that can be attained using any policy. There is also a version of Bellman's equations for the optimal value function:

$$V^*(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V^*(s'). \quad (2)$$

The first term above is the immediate reward as before. The second term is the maximum over all actions a of the expected future sum of discounted rewards we'll get upon after action a

Bellman Equation

We also define a policy $\pi^* : S \mapsto A$ as follows:

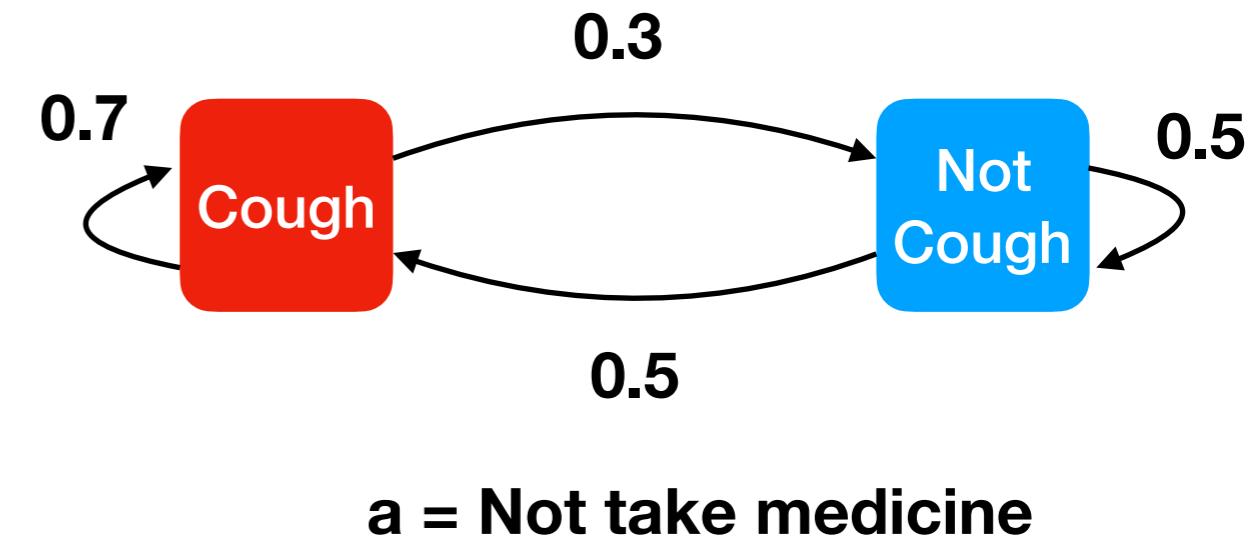
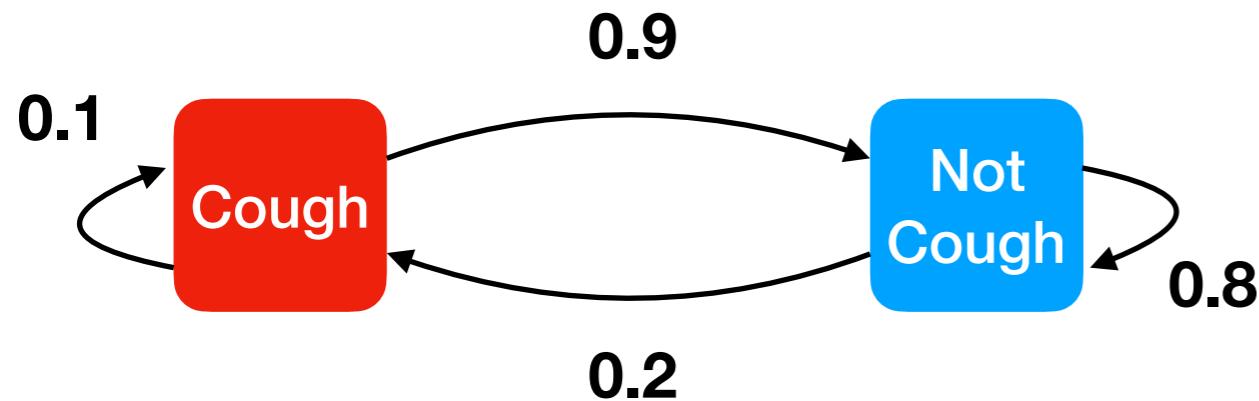
$$\pi^*(s) = \arg \max_{a \in A} \sum_{s' \in S} P_{sa}(s') V^*(s'). \quad (3)$$

Note that $\pi^*(s)$ gives the action a that attains the maximum in the “max” in Equation (2).

It is a fact that for every state s and every policy π , we have

$$V^*(s) = V^{\pi^*}(s) \geq V^\pi(s).$$

What is optimal policy under this example?



Value Iteration

Update value function using Bellman Equations

1. For each state s , initialize $V(s) := 0$.
2. Repeat until convergence {

For every state, update $V(s) := R(s) + \max_{a \in A} \gamma \sum_{s'} P_{sa}(s')V(s')$.

}

Find the optimal policy using

$$\pi^*(s) = \arg \max_{a \in A} \sum_{s' \in S} P_{sa}(s')V^*(s').$$

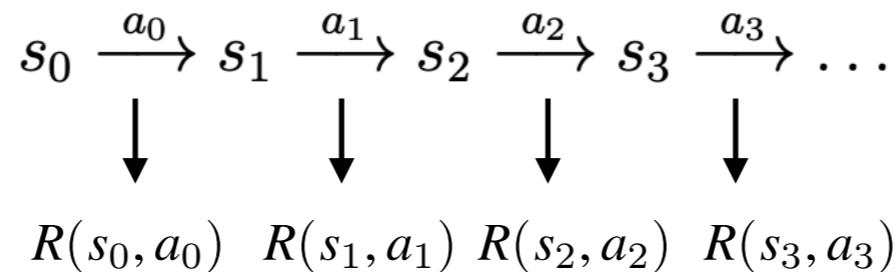
Policy Iteration

Update value function using Bellman Equations

1. Initialize π randomly.
2. Repeat until convergence {
 - (a) Let $V := V^\pi$.
 - (b) For each state s , let $\pi(s) := \arg \max_{a \in A} \sum_{s'} P_{sa}(s')V(s')$.}

It involve solving large linear system.

Action-Value Function



$$G_0 = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots$$

$$V^\pi(s) = E[G_0 | s_0 = s]$$

$$q_\pi(s, a) = E[G_0 | s_0 = s, a_0 = a]$$

We have the Bellman equation

$$V^\pi(s) = R(s) + \gamma \sum P_{s\pi(s)}(s') V^\pi(s')$$

$$q_\pi(s, a) = R(s) + \gamma \sum P_{sa}(s') V^\pi(s')$$

$$V^\pi(s) = q_\pi(s, \pi(s))$$

Action-Value Function

We also define the **optimal value function** according to

$$V^*(s) = \max_{\pi} V^{\pi}(s). \quad (1)$$

In other words, this is the best possible expected sum of discounted rewards that can be attained using any policy. There is also a version of Bellman's equations for the optimal value function:

$$V^*(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V^*(s'). \quad (2)$$

Optimal Value function

$$q_*(s, a) = \max_{\pi} q_{\pi}(s, a)$$

$$V^{\pi}(s) = q_{\pi}(s, \pi(s))$$

Q-Learning

Updating Rule

$$Q(s, a) = (1 - \alpha)Q(s, a) + \gamma(R(s') + \gamma \max_{a' \in A} Q(s', a'))$$

Model-Free Reinforcement Learning

Last Time: Estimate the value function of an known MDP

Value iteration / Policy iteration

This Time: Estimate the value function of an unknown MDP

Model-Free Reinforcement Learning

Last Time: Estimate the value function of an known MDP

Value iteration / Policy iteration

This Time: Estimate the value function of an unknown MDP

Model-free: no knowledge of MDP transitions / rewards

Monte-Carlo Reinforcement Learning

- MC methods learn directly from episodes of experience
- MC is *model-free*: no knowledge of MDP transitions / rewards
- MC learns from *complete* episodes: no bootstrapping
- MC uses the simplest possible idea: value = mean return
- Caveat: can only apply MC to *episodic* MDPs
 - All episodes must terminate

Monte-Carlo Reinforcement Learning

- Goal: learn v_π from episodes of experience under policy π

$$S_1, A_1, R_2, \dots, S_k \sim \pi$$

- Recall that the *return* is the total discounted reward:

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T$$

- Recall that the value function is the expected return:

$$v_\pi(s) = \mathbb{E}_\pi [G_t \mid S_t = s]$$

- Monte-Carlo policy evaluation uses *empirical mean* return instead of *expected* return

Monte-Carlo Reinforcement Learning

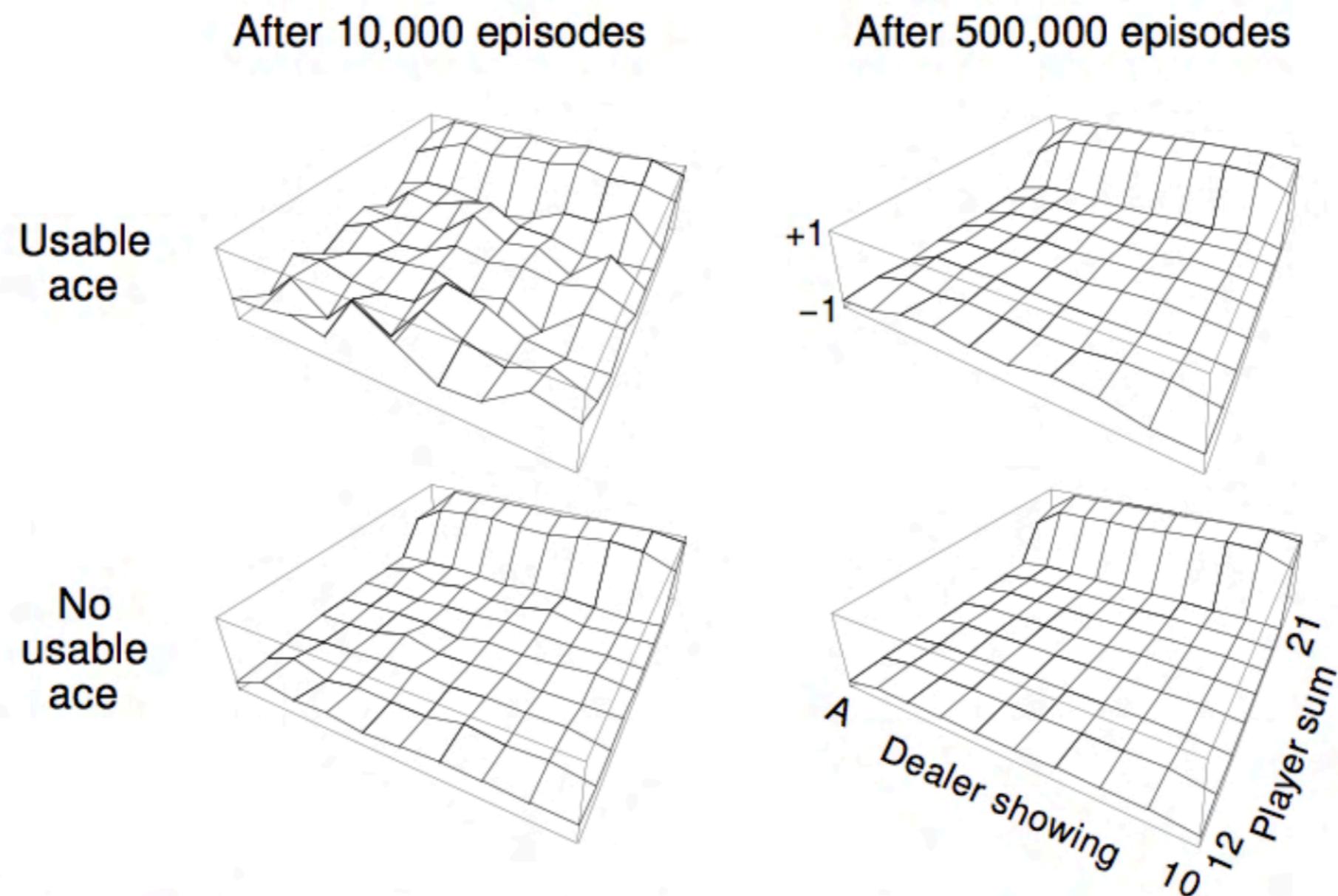
- To evaluate state s
- **Every** time-step t that state s is visited in an episode,
- Increment counter $N(s) \leftarrow N(s) + 1$
- Increment total return $S(s) \leftarrow S(s) + G_t$
- Value is estimated by mean return $V(s) = S(s)/N(s)$
- Again, $V(s) \rightarrow v_\pi(s)$ as $N(s) \rightarrow \infty$

Blackjack Example

- States (200 of them):
 - Current sum (12-21)
 - Dealer's showing card (ace-10)
 - Do I have a "useable" ace? (yes-no)
- Action **stick**: Stop receiving cards (and terminate)
- Action **twist**: Take another card (no replacement)
- Reward for **stick**:
 - +1 if sum of cards > sum of dealer cards
 - 0 if sum of cards = sum of dealer cards
 - -1 if sum of cards < sum of dealer cards
- Reward for **twist**:
 - -1 if sum of cards > 21 (and terminate)
 - 0 otherwise
- Transitions: automatically **twist** if sum of cards < 12



Blackjack Example



Policy: **stick** if sum of cards ≥ 20 , otherwise **twist**

Incremental Mean

The mean μ_1, μ_2, \dots of a sequence x_1, x_2, \dots can be computed incrementally,

$$\begin{aligned}\mu_k &= \frac{1}{k} \sum_{j=1}^k x_j \\ &= \frac{1}{k} \left(x_k + \sum_{j=1}^{k-1} x_j \right) \\ &= \frac{1}{k} (x_k + (k-1)\mu_{k-1}) \\ &= \mu_{k-1} + \frac{1}{k} (x_k - \mu_{k-1})\end{aligned}$$

Incremental Mean

- Update $V(s)$ incrementally after episode $S_1, A_1, R_2, \dots, S_T$
- For each state S_t with return G_t

$$N(S_t) \leftarrow N(S_t) + 1$$

$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)} (G_t - V(S_t))$$

- In non-stationary problems, it can be useful to track a running mean, i.e. forget old episodes.

$$V(S_t) \leftarrow V(S_t) + \alpha (G_t - V(S_t))$$

Temporal-Difference Learning

- TD methods learn directly from episodes of experience
- TD is *model-free*: no knowledge of MDP transitions / rewards
- TD learns from *incomplete* episodes, by *bootstrapping*
- TD updates a guess towards a guess

MC vs TD

- Goal: learn v_π online from experience under policy π
- Incremental every-visit Monte-Carlo
 - Update value $V(S_t)$ toward *actual* return G_t

$$V(S_t) \leftarrow V(S_t) + \alpha (G_t - V(S_t))$$

- Simplest temporal-difference learning algorithm: TD(0)
 - Update value $V(S_t)$ toward *estimated* return $R_{t+1} + \gamma V(S_{t+1})$

$$V(S_t) \leftarrow V(S_t) + \alpha (R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

- $R_{t+1} + \gamma V(S_{t+1})$ is called the *TD target*
- $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called the *TD error*

MC vs TD

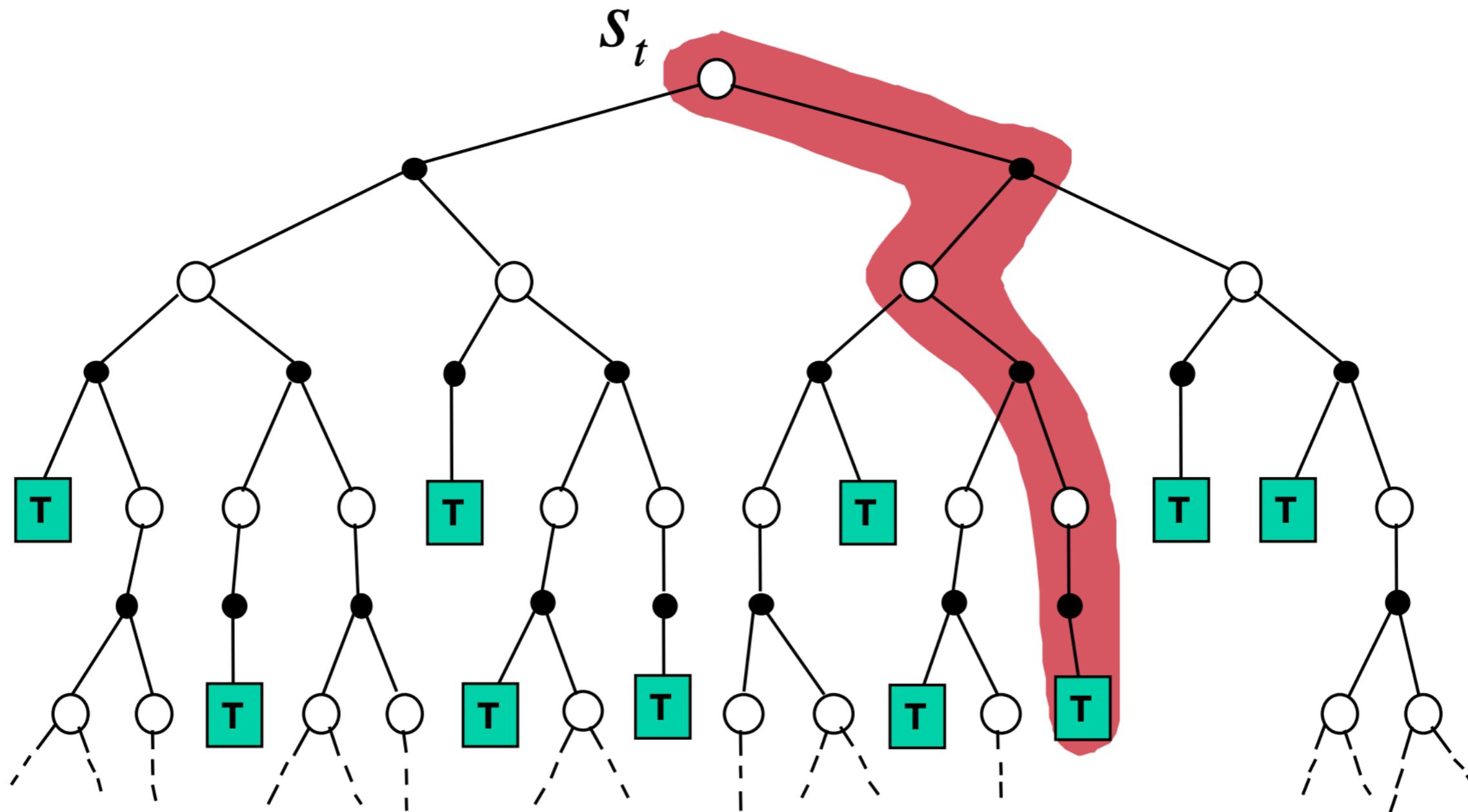
- TD can learn *before* knowing the final outcome
 - TD can learn online after every step
 - MC must wait until end of episode before return is known
- TD can learn *without* the final outcome
 - TD can learn from incomplete sequences
 - MC can only learn from complete sequences
 - TD works in continuing (non-terminating) environments
 - MC only works for episodic (terminating) environments

Bias/Variance Trade-Off

- Return $G_t = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T$ is *unbiased* estimate of $v_\pi(S_t)$
- True TD target $R_{t+1} + \gamma v_\pi(S_{t+1})$ is *unbiased* estimate of $v_\pi(S_t)$
- TD target $R_{t+1} + \gamma V(S_{t+1})$ is *biased* estimate of $v_\pi(S_t)$
- TD target is much lower variance than the return:
 - Return depends on *many* random actions, transitions, rewards
 - TD target depends on *one* random action, transition, reward

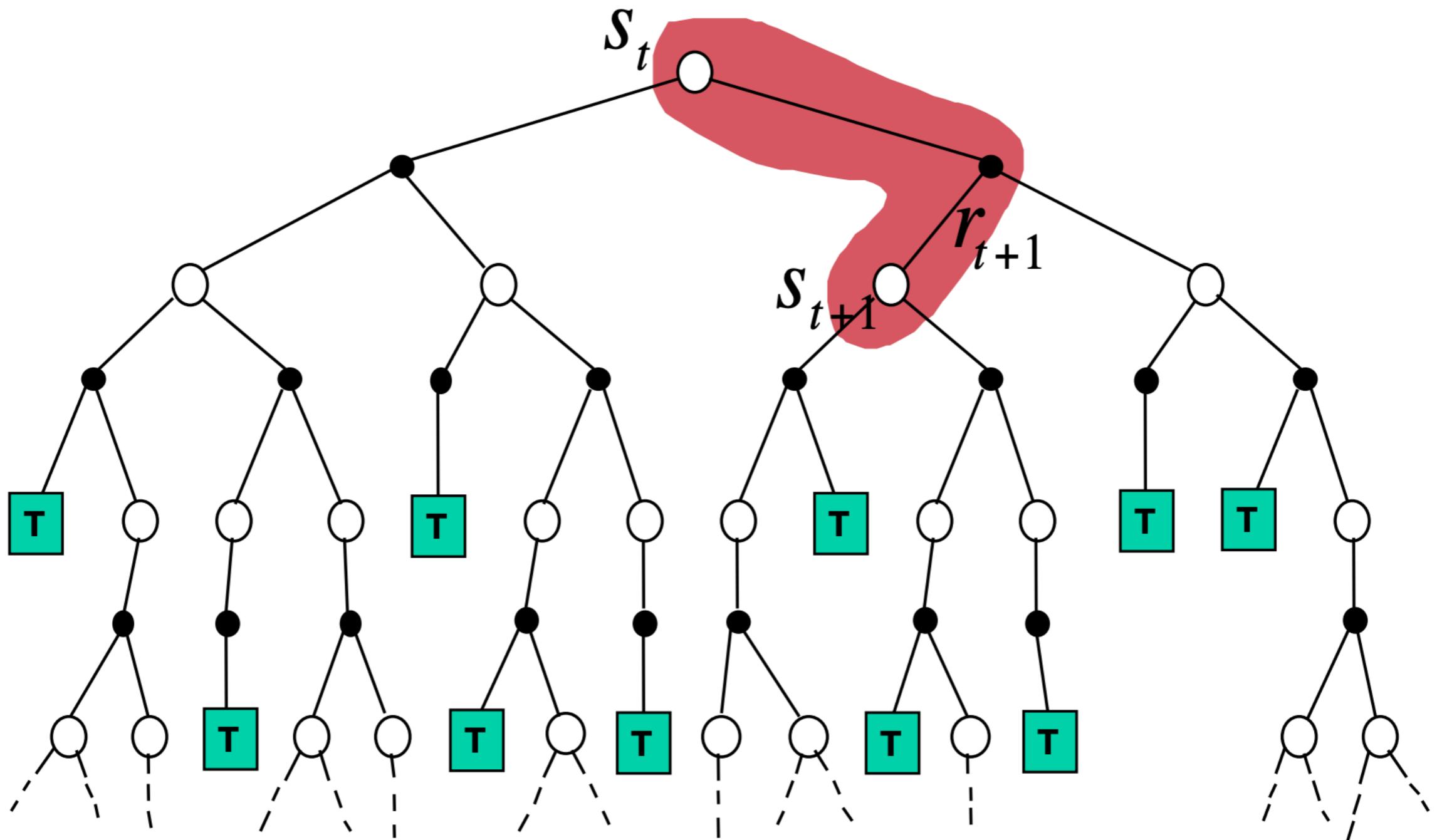
Monte-Carlo Backup

$$V(S_t) \leftarrow V(S_t) + \alpha (G_t - V(S_t))$$



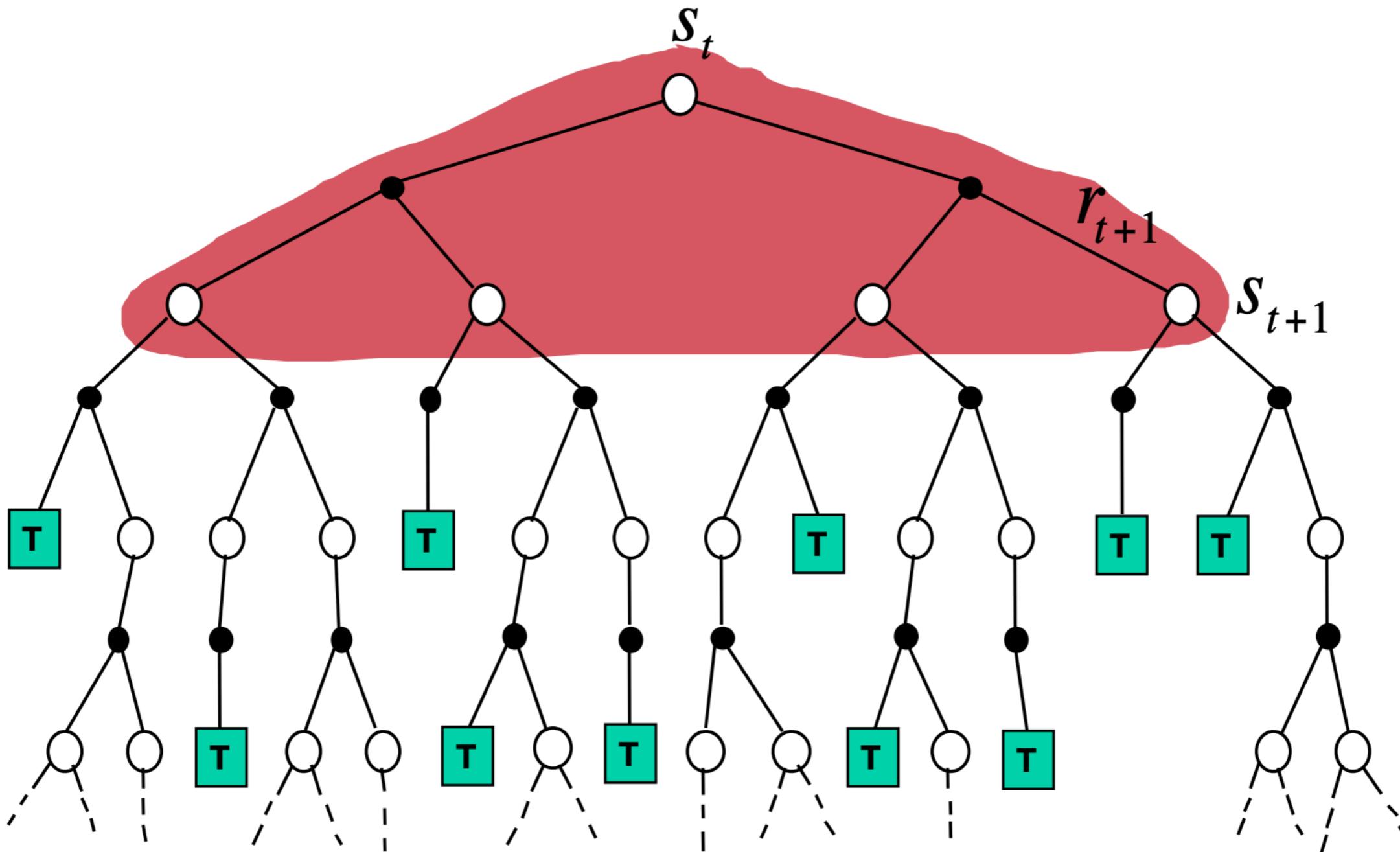
TD Backup

$$V(S_t) \leftarrow V(S_t) + \alpha (R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$



Value Iteration

$$V(S_t) \leftarrow \mathbb{E}_\pi [R_{t+1} + \gamma V(S_{t+1})]$$



Model-Free Control

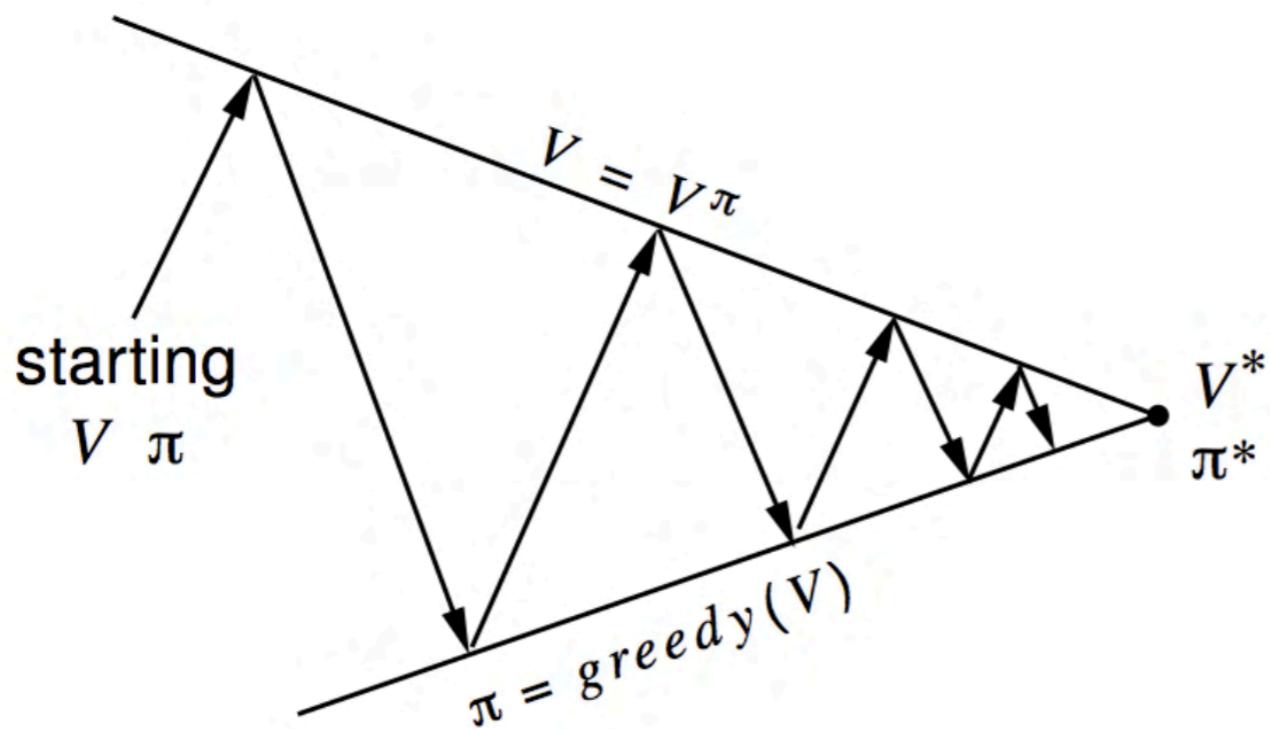
Model-Free Prediction:

Estimate the value function of an unknown MDP

Model-Free Control:

Optimize the value function of an unknown MDP

Model-Free Control

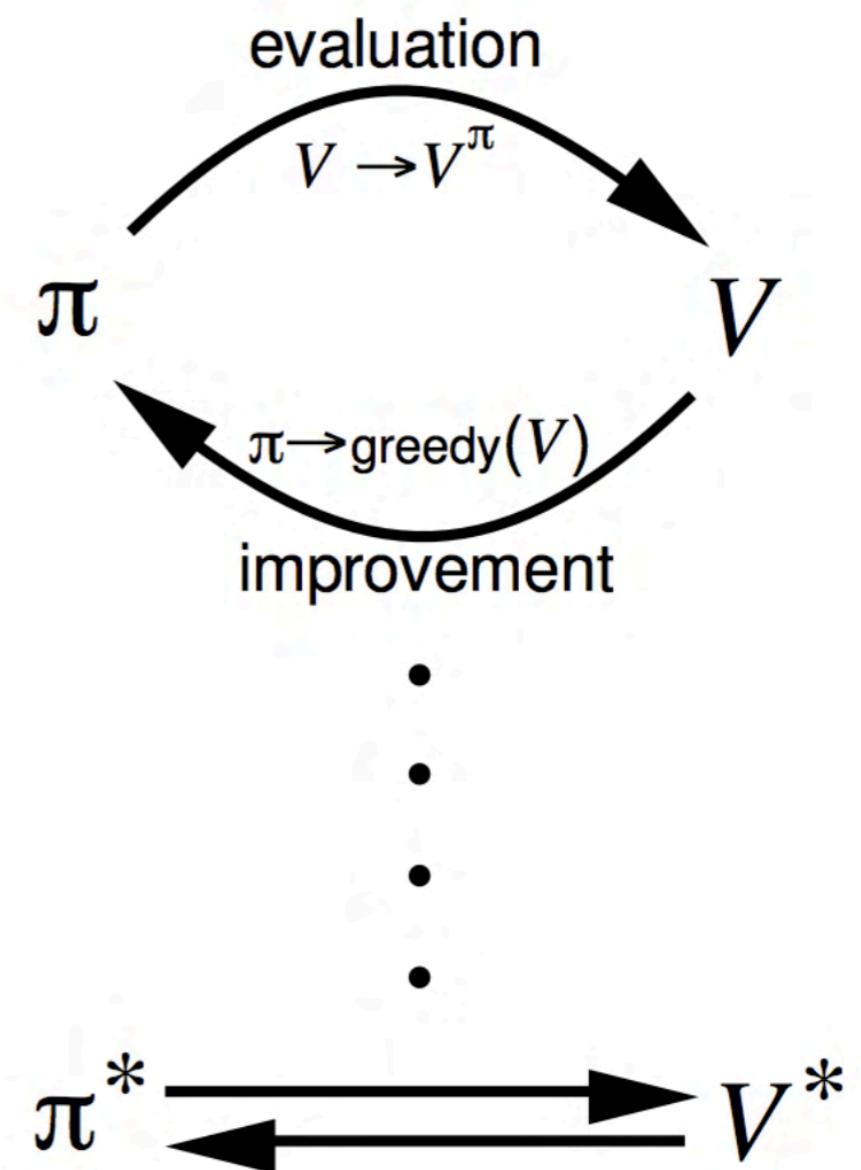


Policy evaluation Estimate v_π

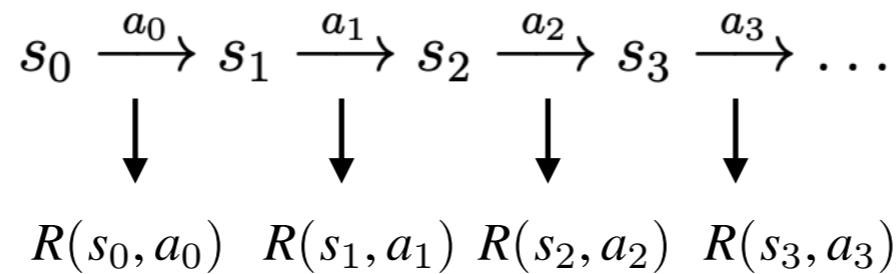
e.g. Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$

e.g. Greedy policy improvement



Action-Value Function



$$G_0 = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots$$

$$V^\pi(s) = E[G_0 | s_0 = s]$$

$$q_\pi(s, a) = E[G_0 | s_0 = s, a_0 = a]$$

We have the Bellman equation

$$V^\pi(s) = R(s) + \gamma \sum P_{s\pi(s)}(s') V^\pi(s')$$

$$q_\pi(s, a) = R(s) + \gamma \sum P_{sa}(s') V^\pi(s')$$

$$V^\pi(s) = q_\pi(s, \pi(s))$$

Action-Value Function

We also define the **optimal value function** according to

$$V^*(s) = \max_{\pi} V^{\pi}(s). \quad (1)$$

In other words, this is the best possible expected sum of discounted rewards that can be attained using any policy. There is also a version of Bellman's equations for the optimal value function:

$$V^*(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V^*(s'). \quad (2)$$

Optimal Value function

$$q_*(s, a) = \max_{\pi} q_{\pi}(s, a)$$

$$V^{\pi}(s) = q_{\pi}(s, \pi(s))$$

Model-Free Control

Greedy policy improvement over $V(s)$ requires known MDP

$$\pi^*(s) = \arg \max_{a \in A} \sum_{s' \in S} P_{sa}(s') V^*(s').$$

Greedy policy improvement over $Q(s,a)$ requires known MDP

$$\pi^*(s) = \operatorname{argmax}_a q^*(s, a)$$

Example of Greedy Action Selection



- There are two doors in front of you.
- You open the left door and get reward 0
 $V(\text{left}) = 0$
- You open the right door and get reward +1
 $V(\text{right}) = +1$
- You open the right door and get reward +3
 $V(\text{right}) = +2$
- You open the right door and get reward +2
 $V(\text{right}) = +2$
- Are you sure you've chosen the best door?

Greedy Exploration

- Simplest idea for ensuring continual exploration
- All m actions are tried with non-zero probability
- With probability $1 - \epsilon$ choose the greedy action
- With probability ϵ choose an action at random

$$\pi(a|s) = \begin{cases} \epsilon/m + 1 - \epsilon & \text{if } a^* = \underset{a \in \mathcal{A}}{\operatorname{argmax}} Q(s, a) \\ \epsilon/m & \text{otherwise} \end{cases}$$

Exploration and Exploitation

For any π , the ϵ -greedy policy π' is always better, i.e., $V^{\pi'}(s) \geq V^\pi(s)$

Monte-Carlo Control

- Sample k th episode using π : $\{S_1, A_1, R_2, \dots, S_T\} \sim \pi$
- For each state S_t and action A_t in the episode,

$$N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)} (G_t - Q(S_t, A_t))$$

- Improve policy based on new action-value function

$$\epsilon \leftarrow 1/k$$

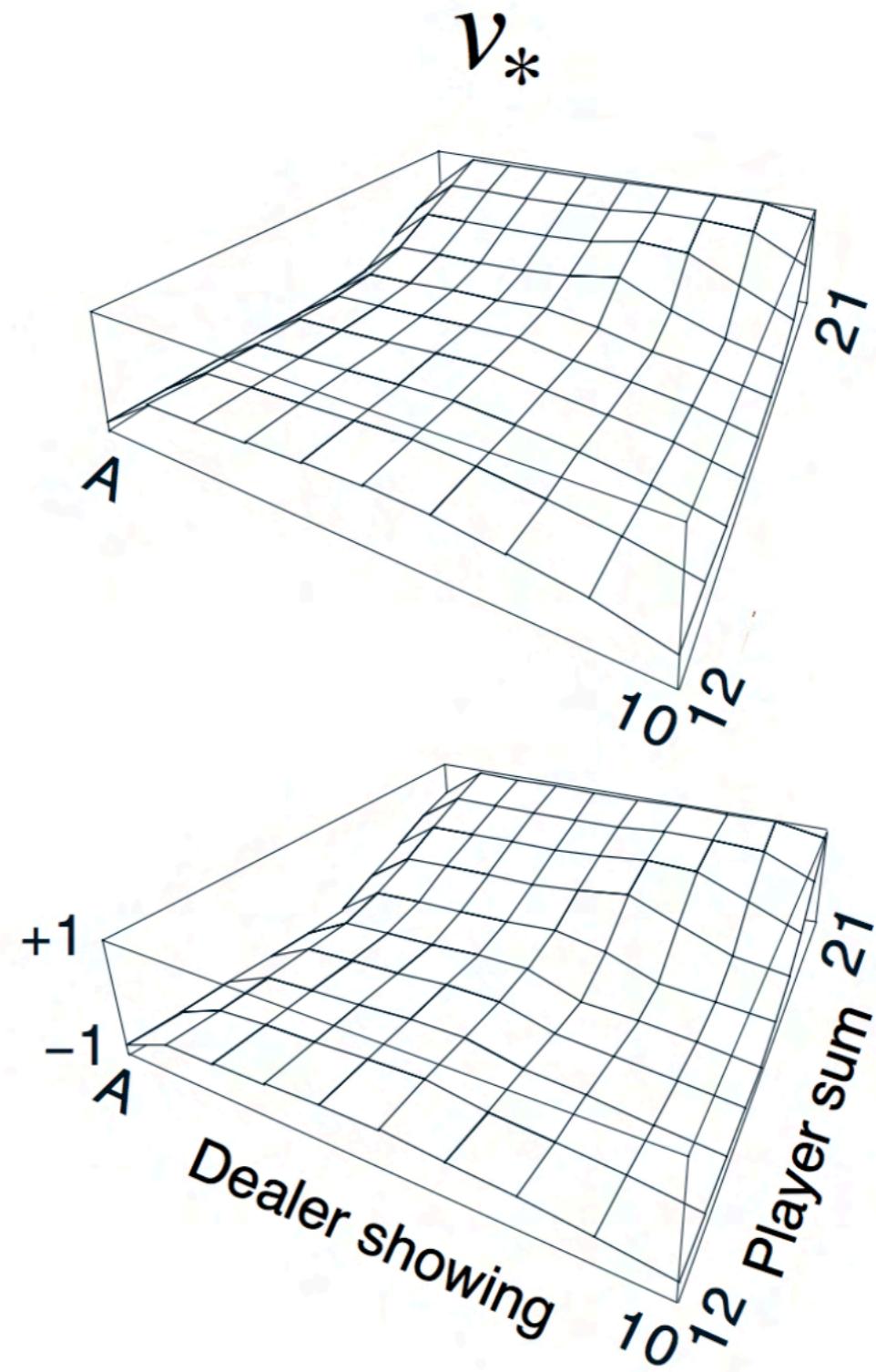
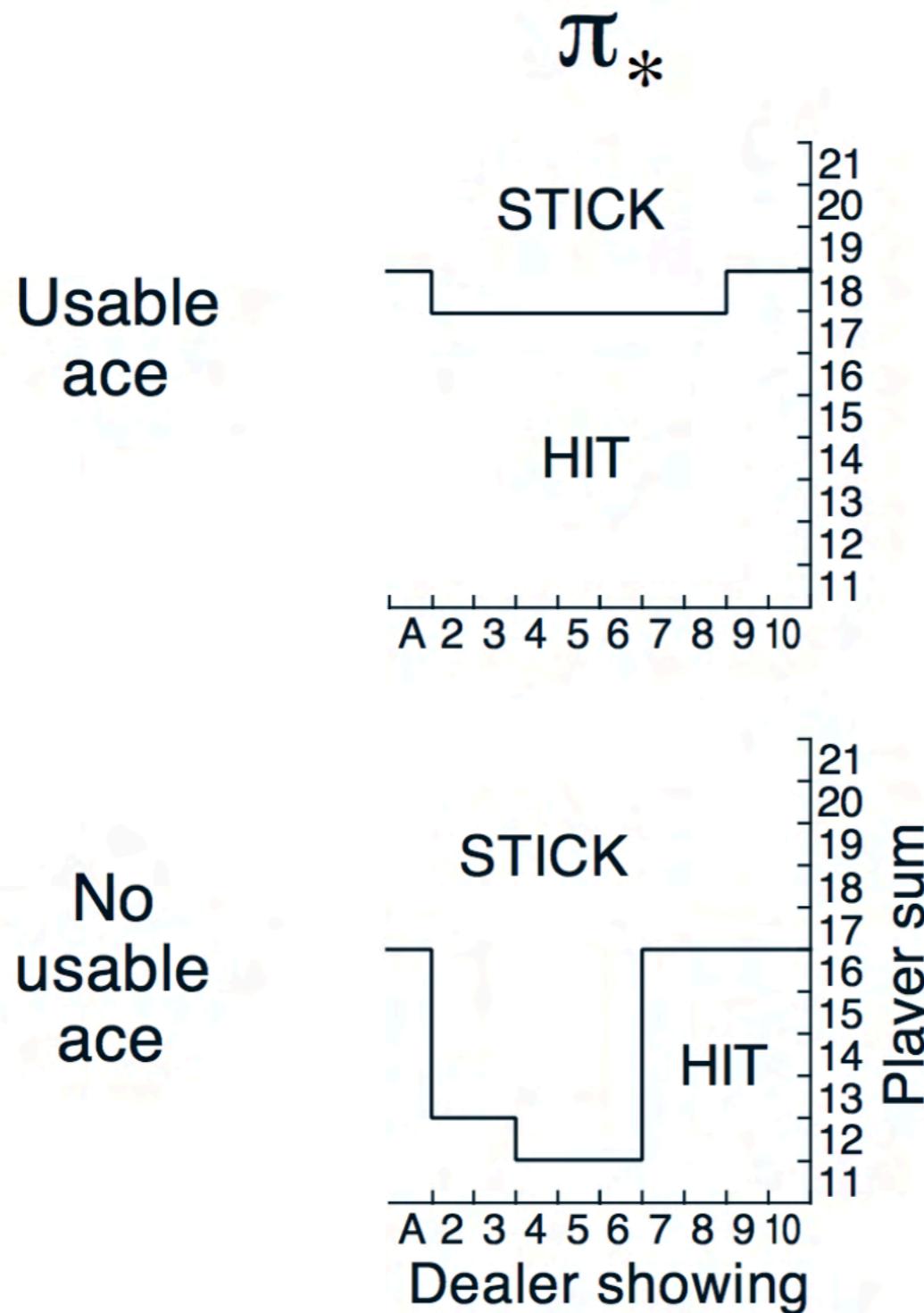
$$\pi \leftarrow \epsilon\text{-greedy}(Q)$$

Back to Blackjack Example

- States (200 of them):
 - Current sum (12-21)
 - Dealer's showing card (ace-10)
 - Do I have a "useable" ace? (yes-no)
- Action **stick**: Stop receiving cards (and terminate)
- Action **twist**: Take another card (no replacement)
- Reward for **stick**:
 - +1 if sum of cards > sum of dealer cards
 - 0 if sum of cards = sum of dealer cards
 - -1 if sum of cards < sum of dealer cards
- Reward for **twist**:
 - -1 if sum of cards > 21 (and terminate)
 - 0 otherwise
- Transitions: automatically **twist** if sum of cards < 12



Back to Blackjack Example



TD Policy Control

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\text{terminal-state}, \cdot) = 0$

Repeat (for each episode):

 Initialize S

 Choose A from S using policy derived from Q (e.g., ε -greedy)

 Repeat (for each step of episode):

 Take action A , observe R, S'

 Choose A' from S' using policy derived from Q (e.g., ε -greedy)

$$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma Q(S', A') - Q(S, A)]$$

$S \leftarrow S'; A \leftarrow A'$;

 until S is terminal

Policy iteration

Q-Learning

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\text{terminal-state}, \cdot) = 0$

Repeat (for each episode):

 Initialize S

 Repeat (for each step of episode):

 Choose A from S using policy derived from Q (e.g., ε -greedy)

 Take action A , observe R, S'

$$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$$

$S \leftarrow S'$;

 until S is terminal

Value iteration

Reinforcement Learning

Junwei Lu



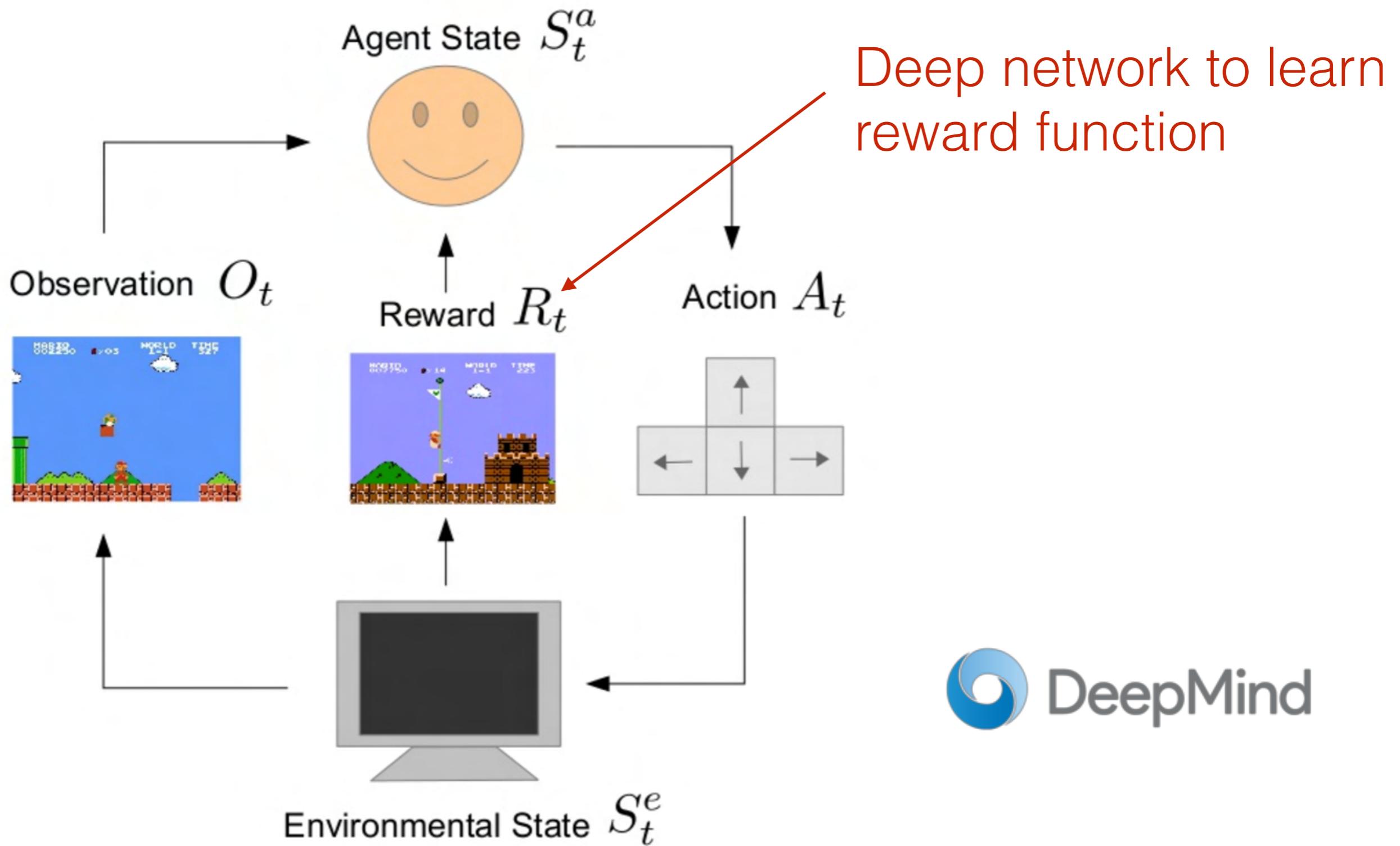
HARVARD
T.H. CHAN

SCHOOL OF PUBLIC HEALTH
Department of Biostatistics



Reinforcement Learning

Reinforcement learning

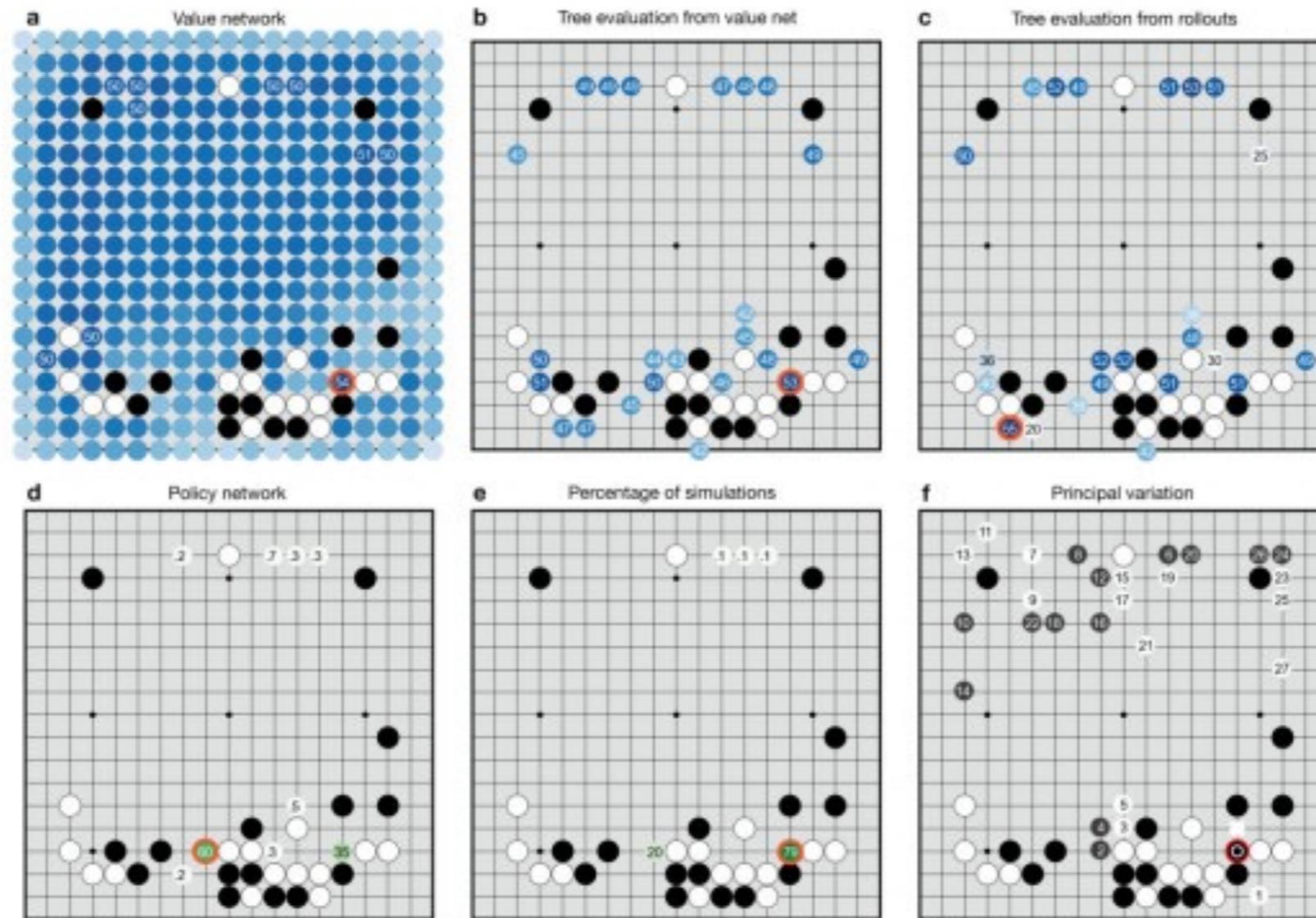


Reinforcement Learning Applications

Reinforcement learning



DeepMind



AlphaGo

Reinforcement Learning Applications

Reinforcement learning

CMU ARTIFICIAL INTELLIGENCE IS TOUGH POKER PLAYER

Libratus builds substantial lead in Brains vs. AI competition

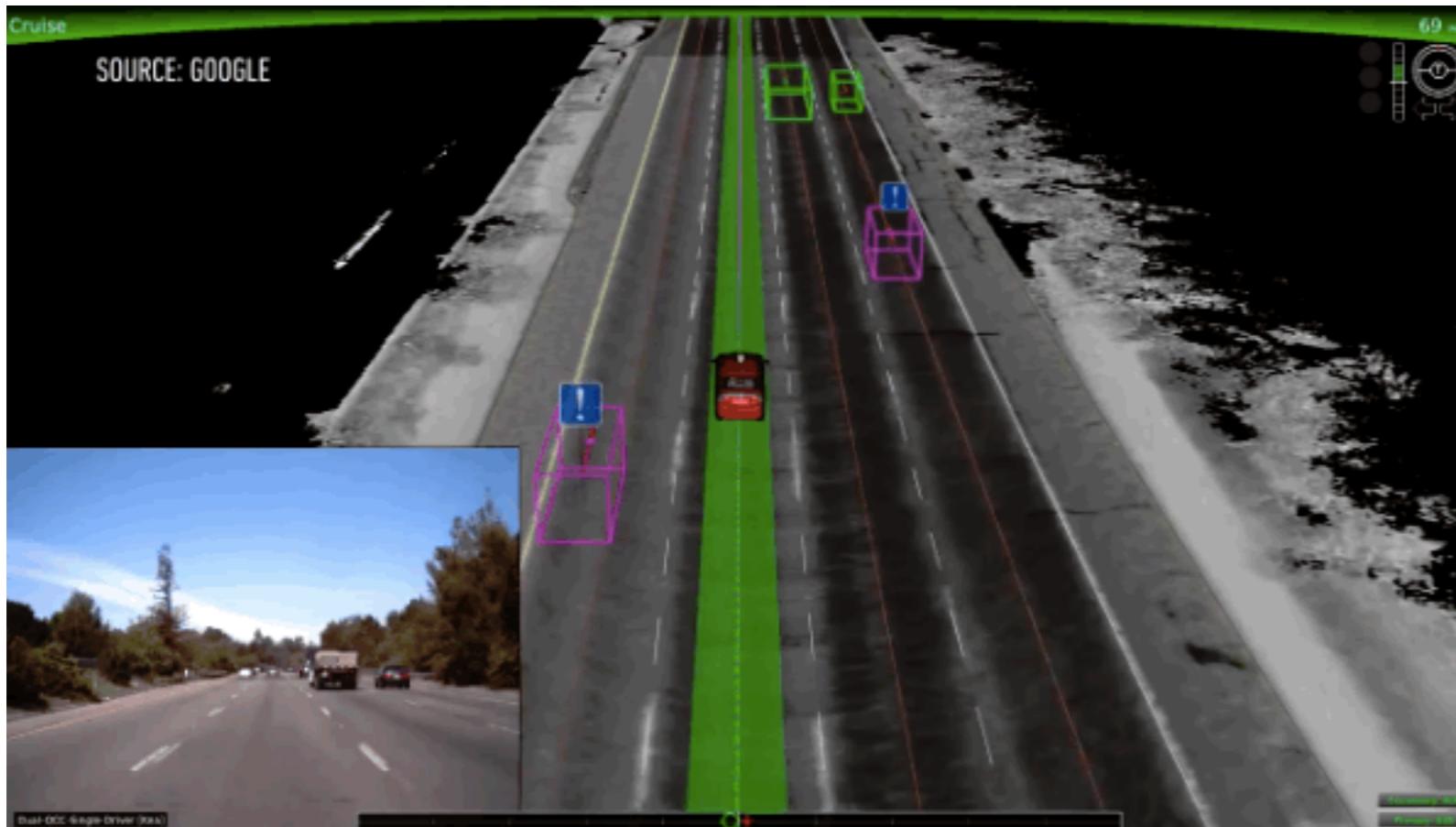
By [Byron Spice](#) (CMU) and [Garrett Allen](#) (Rivers Casino)



Reinforcement Learning Applications

Reinforcement learning + image

Self-driving



Reinforcement Learning in Public Health

Mobile Health



Clinical Trials / Treatment strategy

Reinforcement Learning Applications

Clinical Trials / Treatment strategy

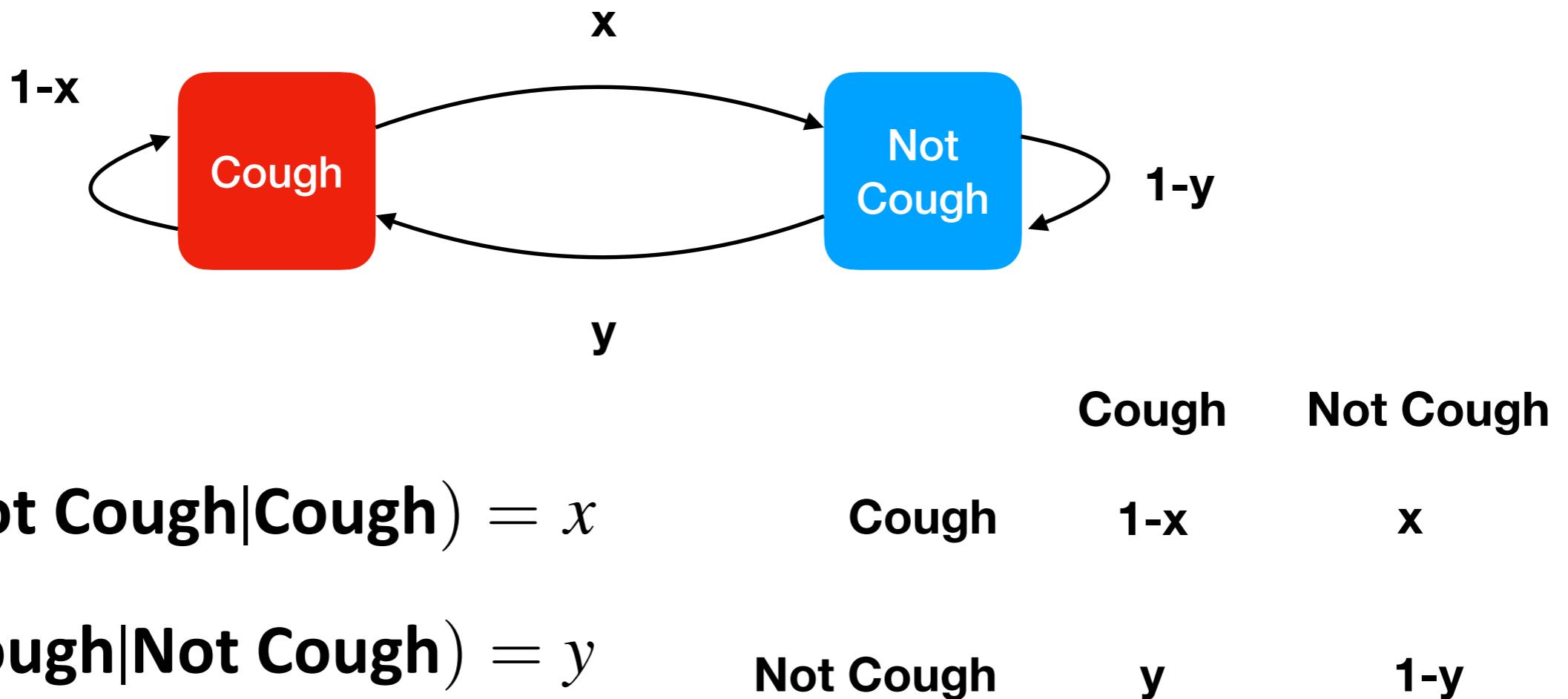
State: Cough / Not Cough

Action: Take medicine / Not take medicine

Reward: Body temperature

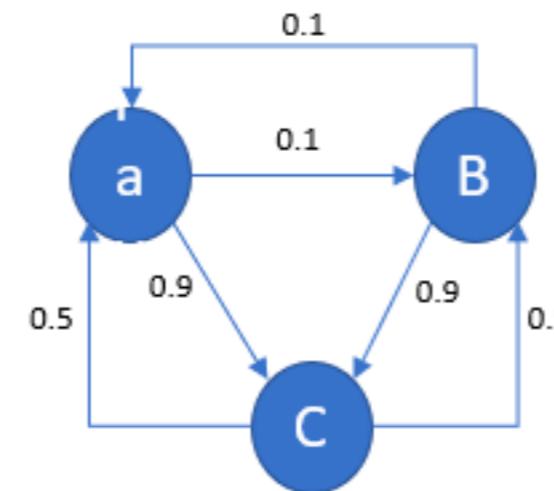
Markov Chain

Markov chain is a system which is in a certain state at each step, with the state changing randomly between steps



Markov Chain

Transition probability



$$P = \begin{bmatrix} 0 & 0.1 & 0.9 \\ 0.1 & 0.0 & 0.9 \\ 0.5 & 0.5 & 0.0 \end{bmatrix}$$

P_s is the state transition probabilities

$P_s(s')$ = the transition probability from s to s'

Markov decision processes

A Markov decision process is a tuple $(S, A, \{P_{sa}\}, \gamma, R)$, where:

S is a set of state, A is a set of actions

- P_{sa} are the state transition probabilities. For each state $s \in S$ and action $a \in A$, P_{sa} is a distribution over the state space. We'll say more about this later, but briefly, P_{sa} gives the distribution over what states we will transition to if we take action a in state s .
- $\gamma \in [0, 1)$ is called the **discount factor**.
- $R : S \times A \mapsto \mathbb{R}$ is the **reward function**. (Rewards are sometimes also written as a function of a state S only, in which case we would have $R : S \mapsto \mathbb{R}$).

State: Cough / Not Cough

Action: Take medicine / Not take medicine

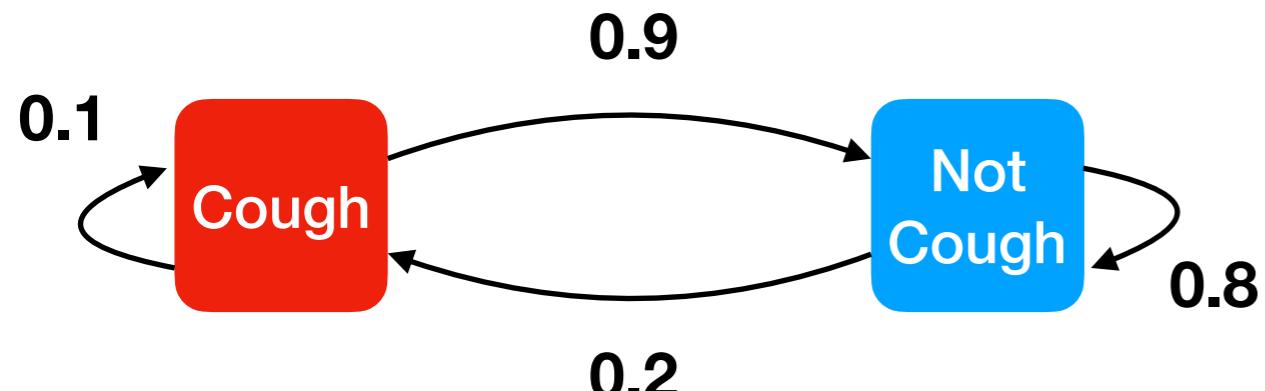
Reward: Body temperature

Markov decision processes

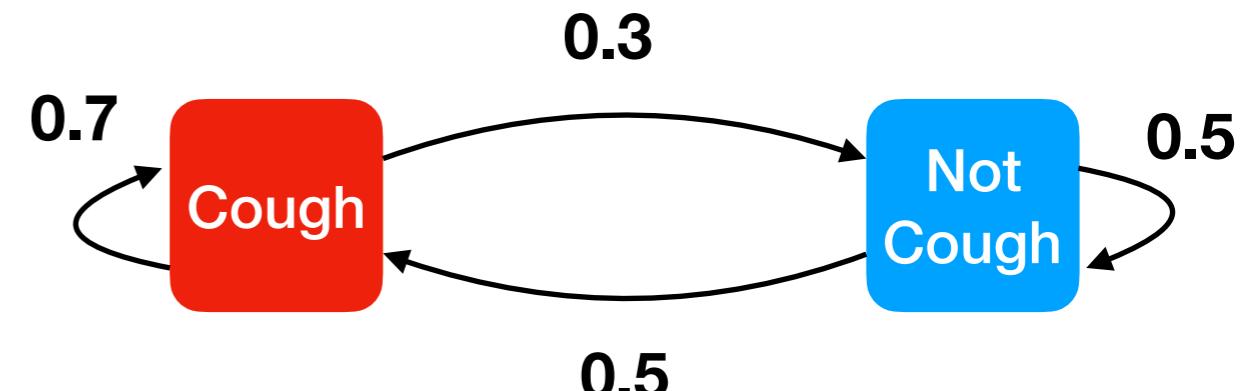
A Markov decision process is a tuple $(S, A, \{P_{sa}\}, R)$, where:

S is a set of states, A is a set of actions

- P_{sa} are the state transition probabilities. For each state $s \in S$ and action $a \in A$, P_{sa} is a distribution over the state space. We'll say more about this later, but briefly, P_{sa} gives the distribution over what states we will transition to if we take action a in state s .
- $\gamma \in [0, 1)$ is called the **discount factor**.
- $R : S \times A \mapsto \mathbb{R}$ is the **reward function**. (Rewards are sometimes also written as a function of a state S only, in which case we would have $R : S \mapsto \mathbb{R}$).



$a = \text{Take medicine}$



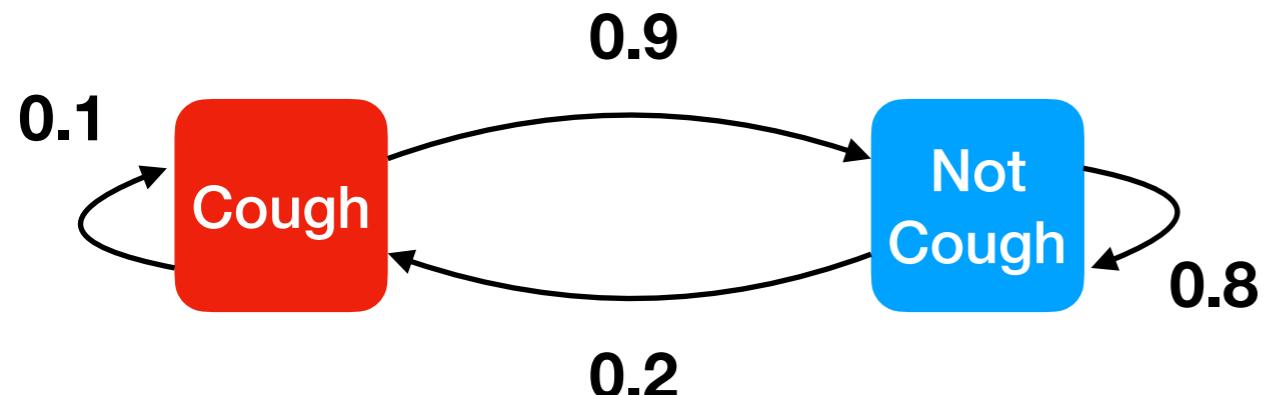
$a = \text{Not take medicine}$

Markov decision processes

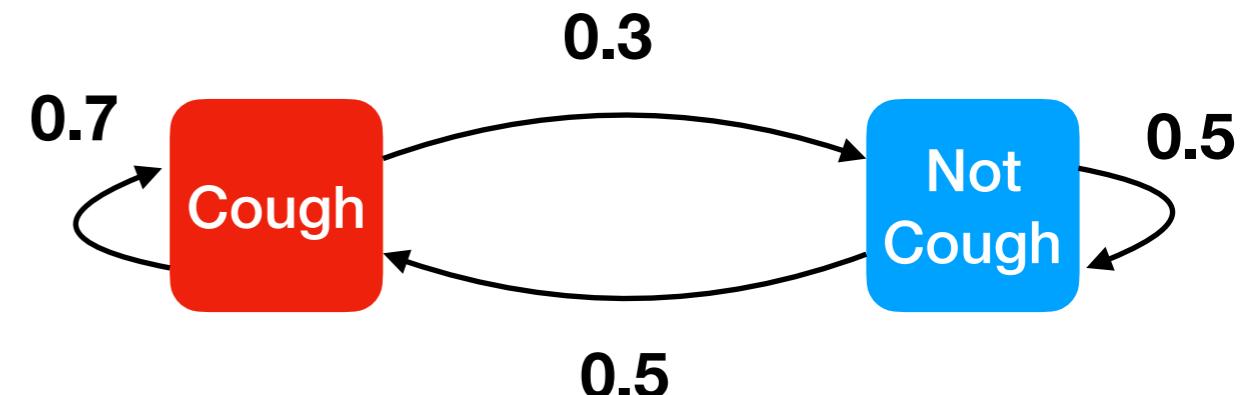
A Markov decision process is a tuple $(S, A, \{P_{sa}\}, R)$, where:

S is a set of states, A is a set of actions

- P_{sa} are the state transition probabilities. For each state $s \in S$ and action $a \in A$, P_{sa} is a distribution over the state space. We'll say more about this later, but briefly, P_{sa} gives the distribution over what states we will transition to if we take action a in state s .
- $\gamma \in [0, 1)$ is called the **discount factor**.
- $R : S \times A \mapsto \mathbb{R}$ is the **reward function**. (Rewards are sometimes also written as a function of a state S only, in which case we would have $R : S \mapsto \mathbb{R}$).



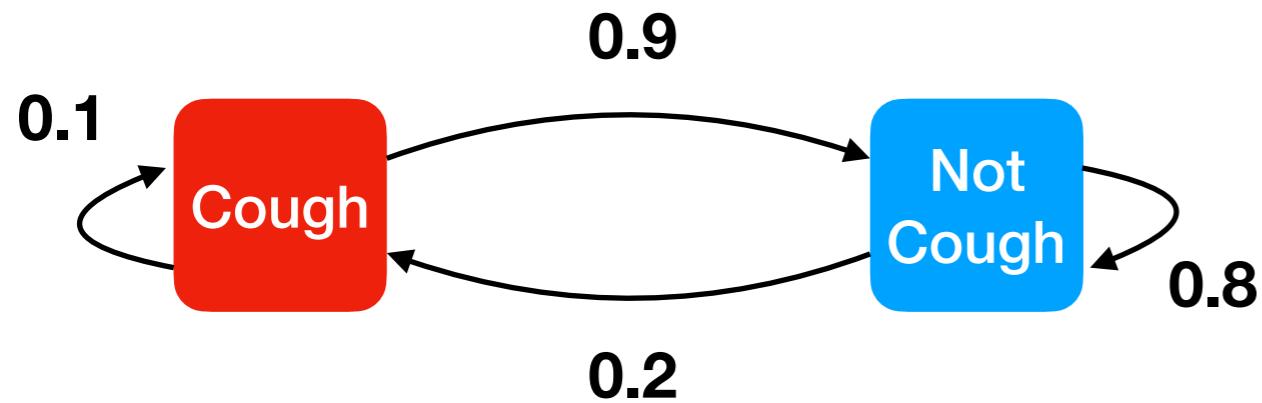
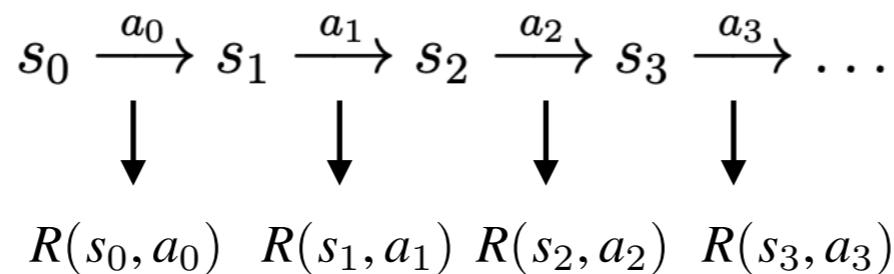
$a = \text{Take medicine}$



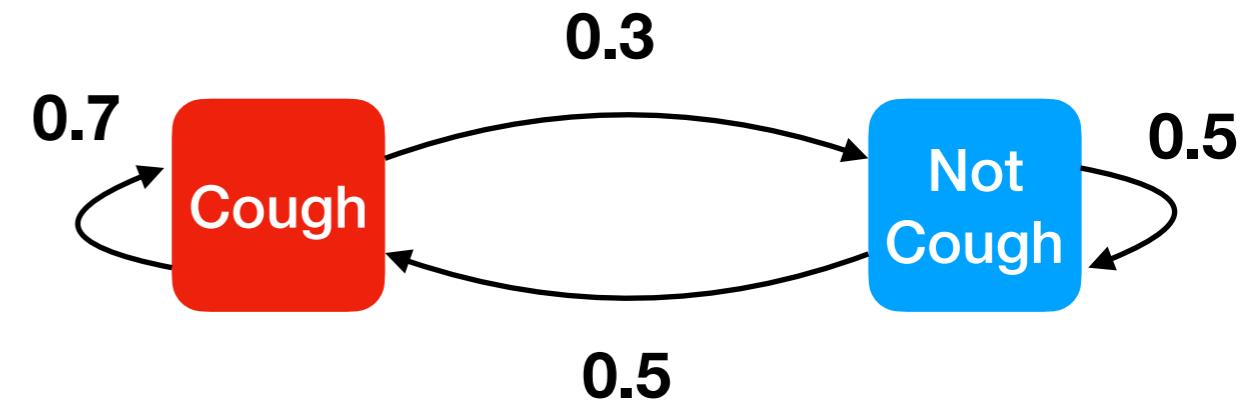
$a = \text{Not take medicine}$

Markov decision processes

The dynamics of an MDP proceeds as follows: We start in some state s_0 , and get to choose some action $a_0 \in A$ to take in the MDP. As a result of our choice, the state of the MDP randomly transitions to some successor state s_1 , drawn according to $s_1 \sim P_{s_0 a_0}$. Then, we get to pick another action a_1 . As a result of this action, the state transitions again, now to some $s_2 \sim P_{s_1 a_1}$. We then pick a_2 , and so on.... Pictorially, we can represent this process as follows:

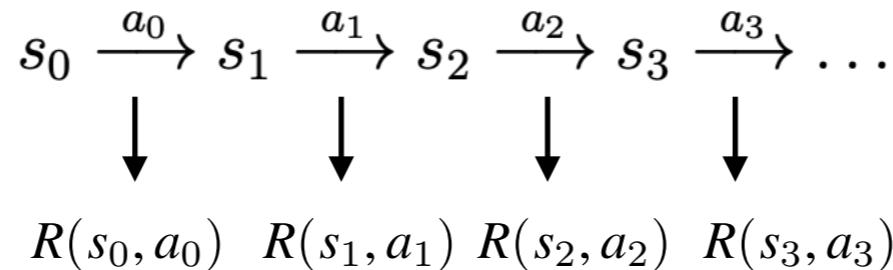


a = Take medicine



a = Not take medicine

Markov decision processes



Upon visiting the sequence of states s_0, s_1, \dots with actions a_0, a_1, \dots , our total payoff is given by

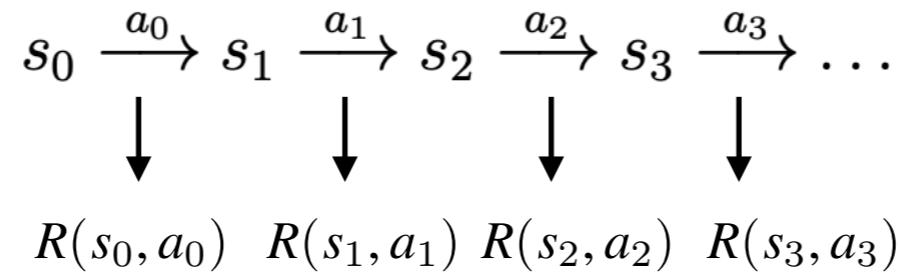
$$R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \dots .$$

Or, when we are writing rewards as a function of the states only, this becomes

$$R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots .$$

For most of our development, we will use the simpler state-rewards $R(s)$, though the generalization to state-action rewards $R(s, a)$ offers no special difficulties.

Markov decision processes



Upon visiting the sequence of states s_0, s_1, \dots with actions a_0, a_1, \dots , our total payoff is given by

$$R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \dots .$$

Or, when we are writing rewards as a function of the states only, this becomes

$$R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots .$$

For most of our development, we will use the simpler state-rewards $R(s)$, though the generalization to state-action rewards $R(s, a)$ offers no special difficulties.

Goal in reinforcement learning is to choose actions over time so as to maximize the expected value of the total payoff:

$$\mathbb{E} [R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots]$$

Markov decision processes

Goal in reinforcement learning is to choose actions over time so as to maximize the expected value of the total payoff:

$$E [R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots]$$

Note that the reward at timestep t is discounted by a factor. Thus, to make this expectation large, we would like to accrue positive rewards as soon as possible (and postpone negative rewards as long as possible).

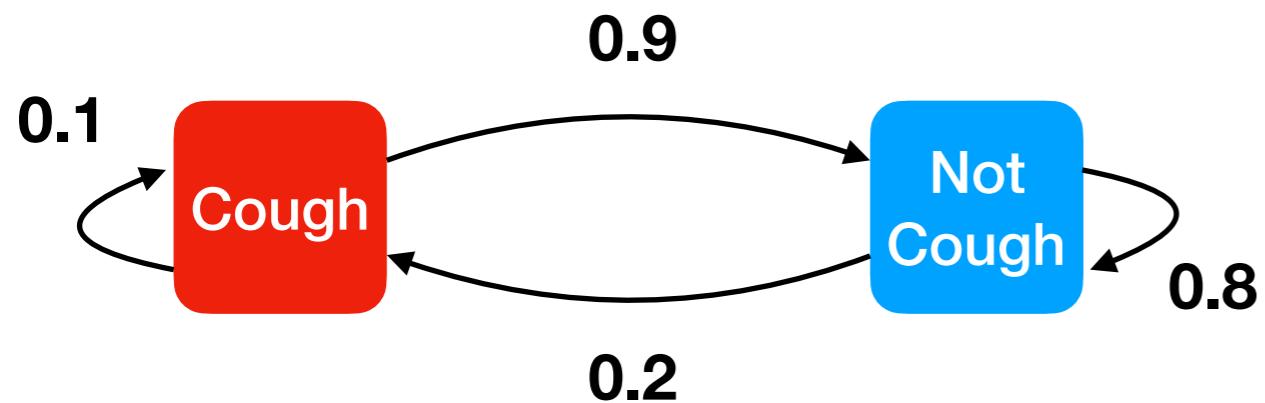
Policy Function

A **policy** is any function $\pi : S \mapsto A$ mapping from the states to the actions. We say that we are **executing** some policy π if, whenever we are in state s , we take action $a = \pi(s)$. We also define the **value function** for a policy π according to

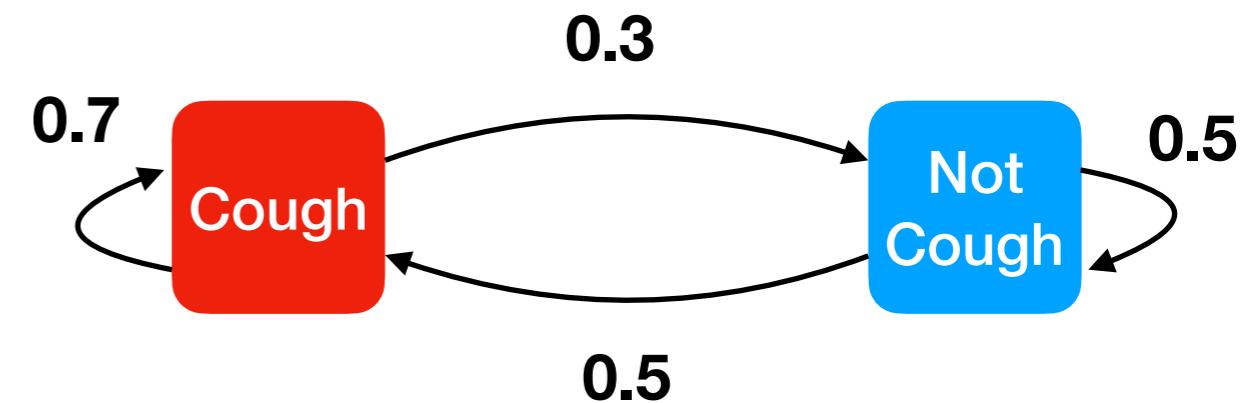
$$V^\pi(s) = \mathbb{E} [R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots | s_0 = s, \pi].$$

$V^\pi(s)$ is simply the expected sum of discounted rewards upon starting in state s , and taking actions according to π .¹

What is policy under this example?



$a = \text{Take medicine}$



$a = \text{Not take medicine}$

Bellman Equation

$$\begin{aligned} V^\pi(s) &= E[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots | s_0 = s] \\ &= E[R(s_0) + \gamma(R(s_1) + \gamma R(s_2) + \dots) | s_0 = s] \\ &= E[R(s_0) + \gamma V^\pi(s_1) | s_0 = s] \end{aligned}$$

The second term is the expected sum of discounted rewards for starting in state s'

$$E_{s' \sim P_{s\pi(s)}} [V^\pi(s')]$$

Given a fixed policy π , its value function V^π satisfies the **Bellman equations**:

$$V^\pi(s) = R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^\pi(s').$$

Bellman's equations can be used to efficiently solve for V^π

Bellman Equation

Given a fixed policy π , its value function V^π satisfies the **Bellman equations**:

$$V^\pi(s) = R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^\pi(s').$$

We have a linear equation

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} \mathcal{R}_1 \\ \vdots \\ \mathcal{R}_n \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{11} & \dots & \mathcal{P}_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$

So we can solve the value function

$$\mathbf{v} = \mathcal{R} + \gamma \mathbf{P} \mathbf{v}$$

$$\mathbf{v} = (I - \gamma \mathbf{P})^{-1} \mathcal{R}$$

Bellman Equation

We also define the **optimal value function** according to

$$V^*(s) = \max_{\pi} V^{\pi}(s). \quad (1)$$

In other words, this is the best possible expected sum of discounted rewards that can be attained using any policy. There is also a version of Bellman's equations for the optimal value function:

$$V^*(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V^*(s'). \quad (2)$$

The first term above is the immediate reward as before. The second term is the maximum over all actions a of the expected future sum of discounted rewards we'll get upon after action a

Bellman Equation

We also define a policy $\pi^* : S \mapsto A$ as follows:

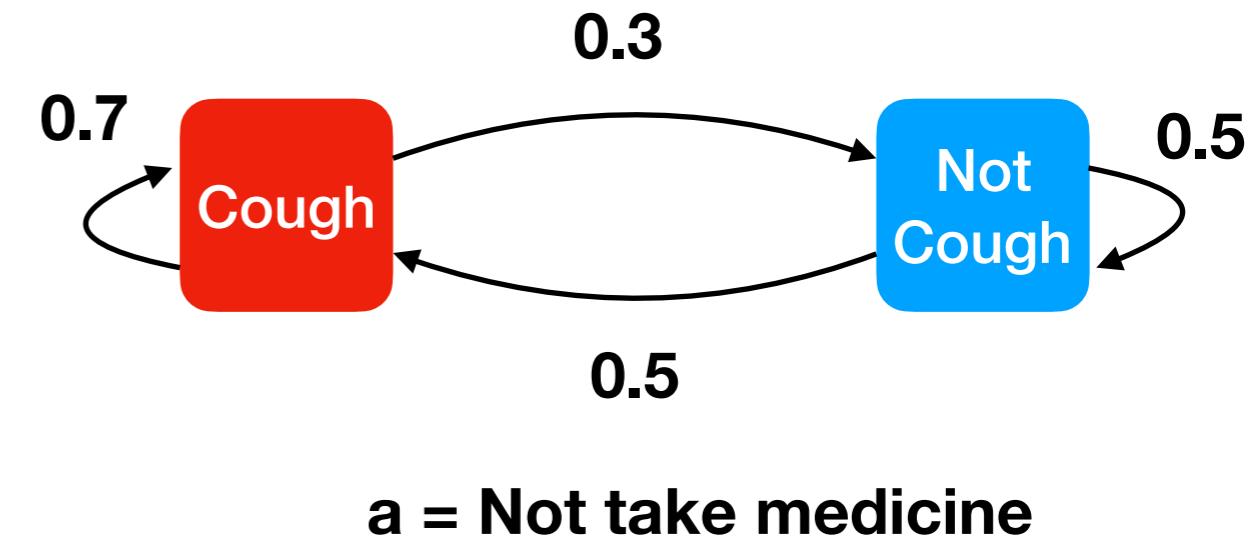
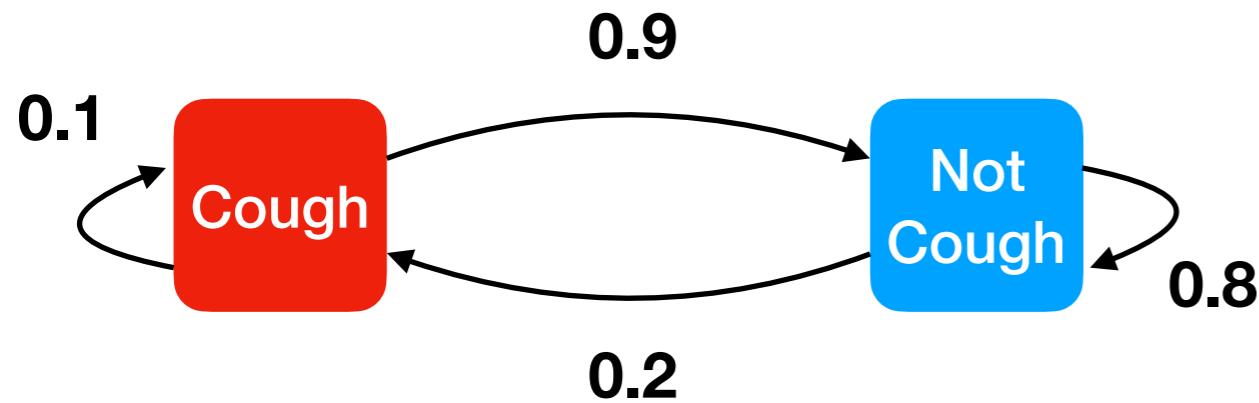
$$\pi^*(s) = \arg \max_{a \in A} \sum_{s' \in S} P_{sa}(s') V^*(s'). \quad (3)$$

Note that $\pi^*(s)$ gives the action a that attains the maximum in the “max” in Equation (2).

It is a fact that for every state s and every policy π , we have

$$V^*(s) = V^{\pi^*}(s) \geq V^\pi(s).$$

What is optimal policy under this example?



Value Iteration

Update value function using Bellman Equations

1. For each state s , initialize $V(s) := 0$.
2. Repeat until convergence {

For every state, update $V(s) := R(s) + \max_{a \in A} \gamma \sum_{s'} P_{sa}(s')V(s')$.

}

Find the optimal policy using

$$\pi^*(s) = \arg \max_{a \in A} \sum_{s' \in S} P_{sa}(s')V^*(s').$$

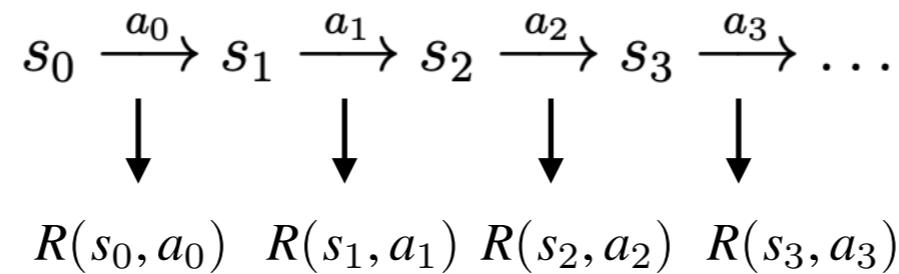
Policy Iteration

Update value function using Bellman Equations

1. Initialize π randomly.
2. Repeat until convergence {
 - (a) Let $V := V^\pi$.
 - (b) For each state s , let $\pi(s) := \arg \max_{a \in A} \sum_{s'} P_{sa}(s')V(s')$.}

It involve solving large linear system.

Action-Value Function



$$G_0 = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots$$

$$V^\pi(s) = E[G_0 | s_0 = s]$$

$$q_\pi(s, a) = E[G_0 | s_0 = s, a_0 = a]$$

We have the Bellman equation

$$V^\pi(s) = R(s) + \gamma \sum P_{s\pi(s)}(s') V^\pi(s')$$

$$q_\pi(s, a) = R(s) + \gamma \sum P_{sa}(s') V^\pi(s')$$

Action-Value Function

We also define the **optimal value function** according to

$$V^*(s) = \max_{\pi} V^{\pi}(s). \quad (1)$$

In other words, this is the best possible expected sum of discounted rewards that can be attained using any policy. There is also a version of Bellman's equations for the optimal value function:

$$V^*(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V^*(s'). \quad (2)$$

Optimal Value function

$$q_*(s, a) = \max_{\pi} q_{\pi}(s, a)$$

Q-Learning

$$Q(s, a) = (1 - \alpha)Q(s, a) + \gamma(R(s') + \gamma \max_{a' \in A} Q(s', a'))$$