From Classification to Quantization: Machine Learning for Communications

JAPAN ADVANCED INSTITUTE OF SCIENCE AND TECHNOLOGY 1990

Brian M. Kurkoski

Japan Advanced Institute of Science and Technology

17 November 2019

Symposium on Future Telecommunications Technologies (SOFTT2019)

Kuala Lumpur, Malaysia

Presentation PDF https://bitly.com/softt2019

Success of Deep Neural Networks

Deep neural networks (DNN) can now match humans on certain object and speech recognition tasks.

DNNs are "programmed" or trained by showing it a large amount of labeled data.

This has successfully replaced the domain knowledge model.

A successful algorithm can correctly recognize an object it had not previously seen.

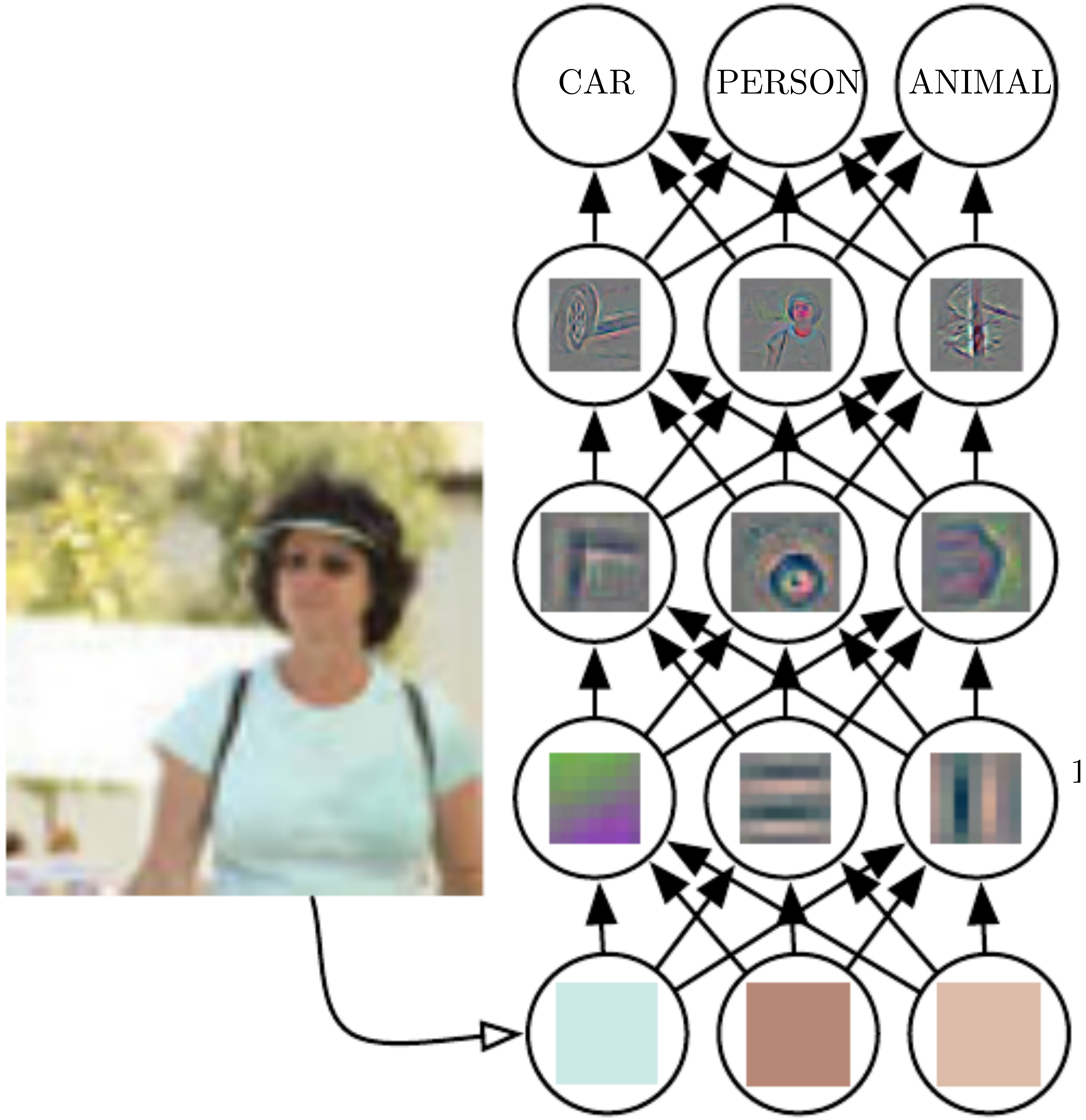
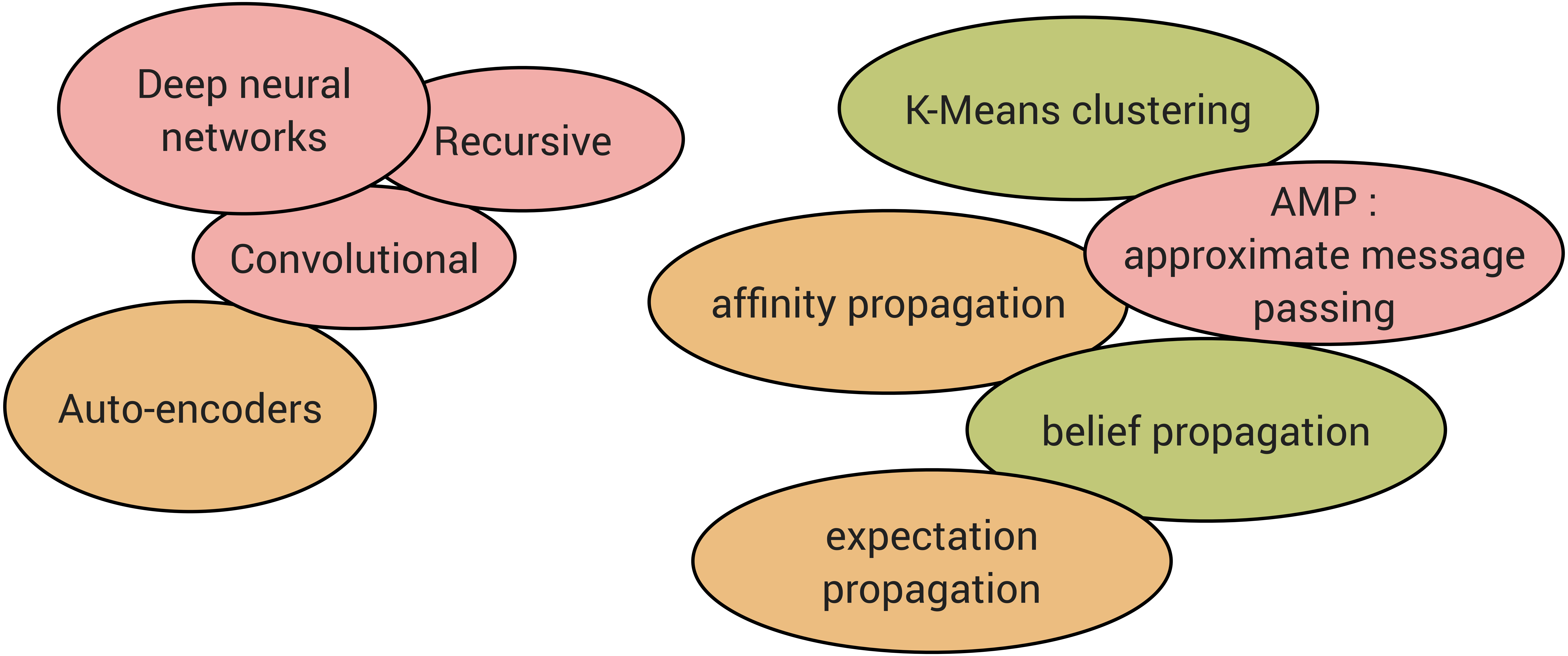


Image: Goodfellow, et al., *Deep learning*.

Machine Learning is More Than DNNs



What is Classification?

A classification algorithm specifies which of k categories some input belongs to

source X

0 →
1 →
2 →
3 →
4 →
5 →
6 →
7 →
8 →
9 →

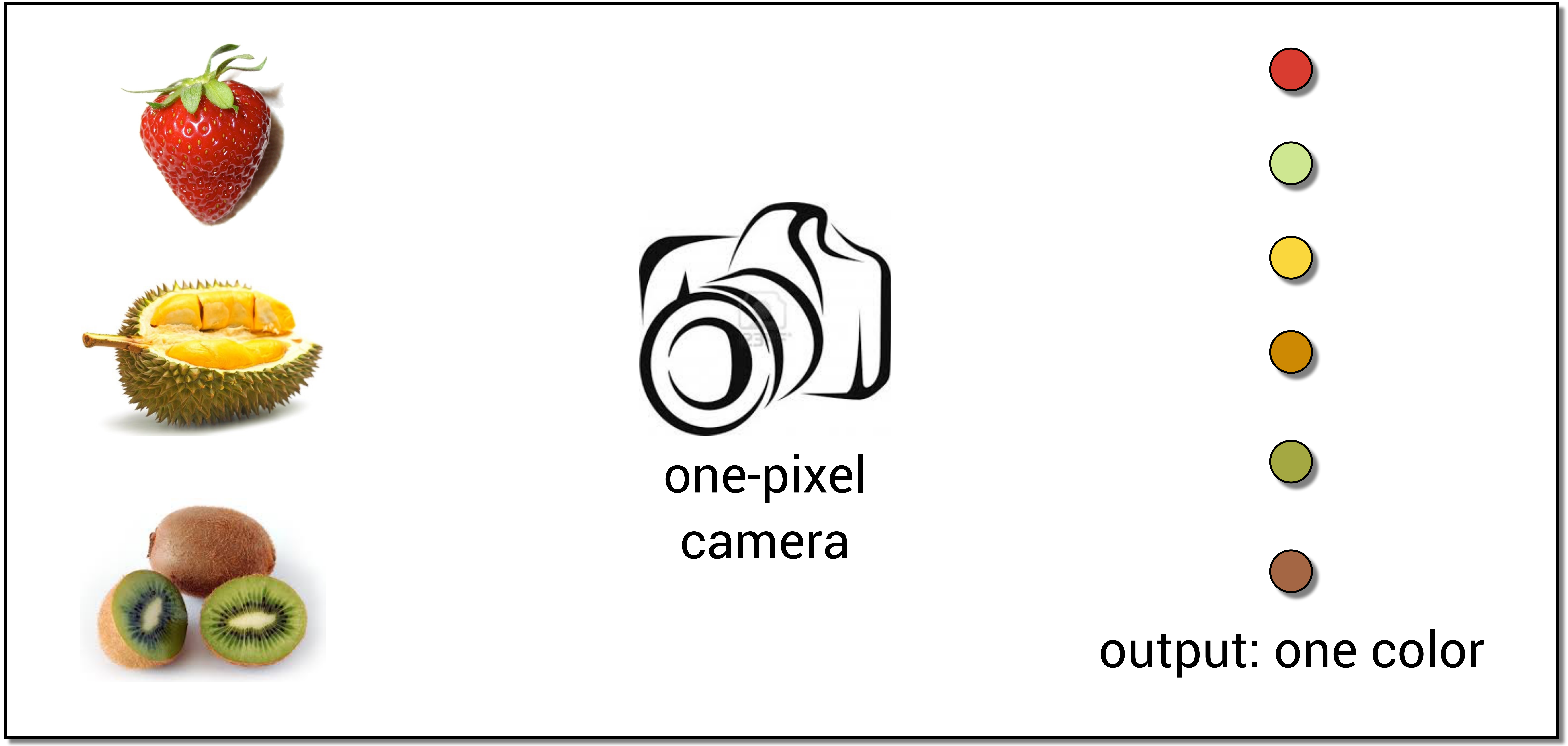
input: noisy observation Y

estimate Z

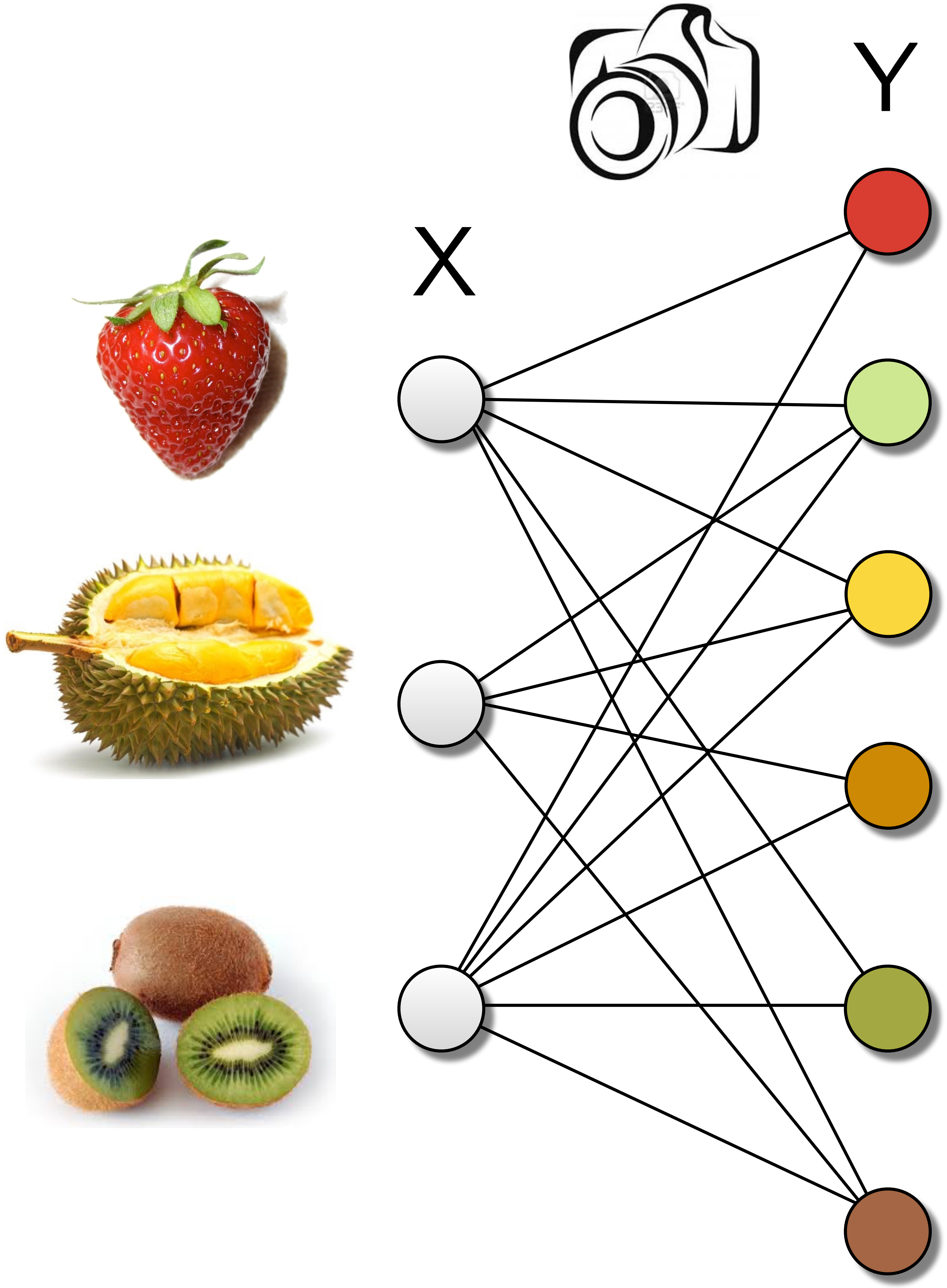
→ 0
→ 1
→ 2
→ 3
→ 4
→ 5
→ 6
→ 7
→ 8
→ 9

Wikimedia/Josef Steppan using MNIST data set

Simple Example of Classification

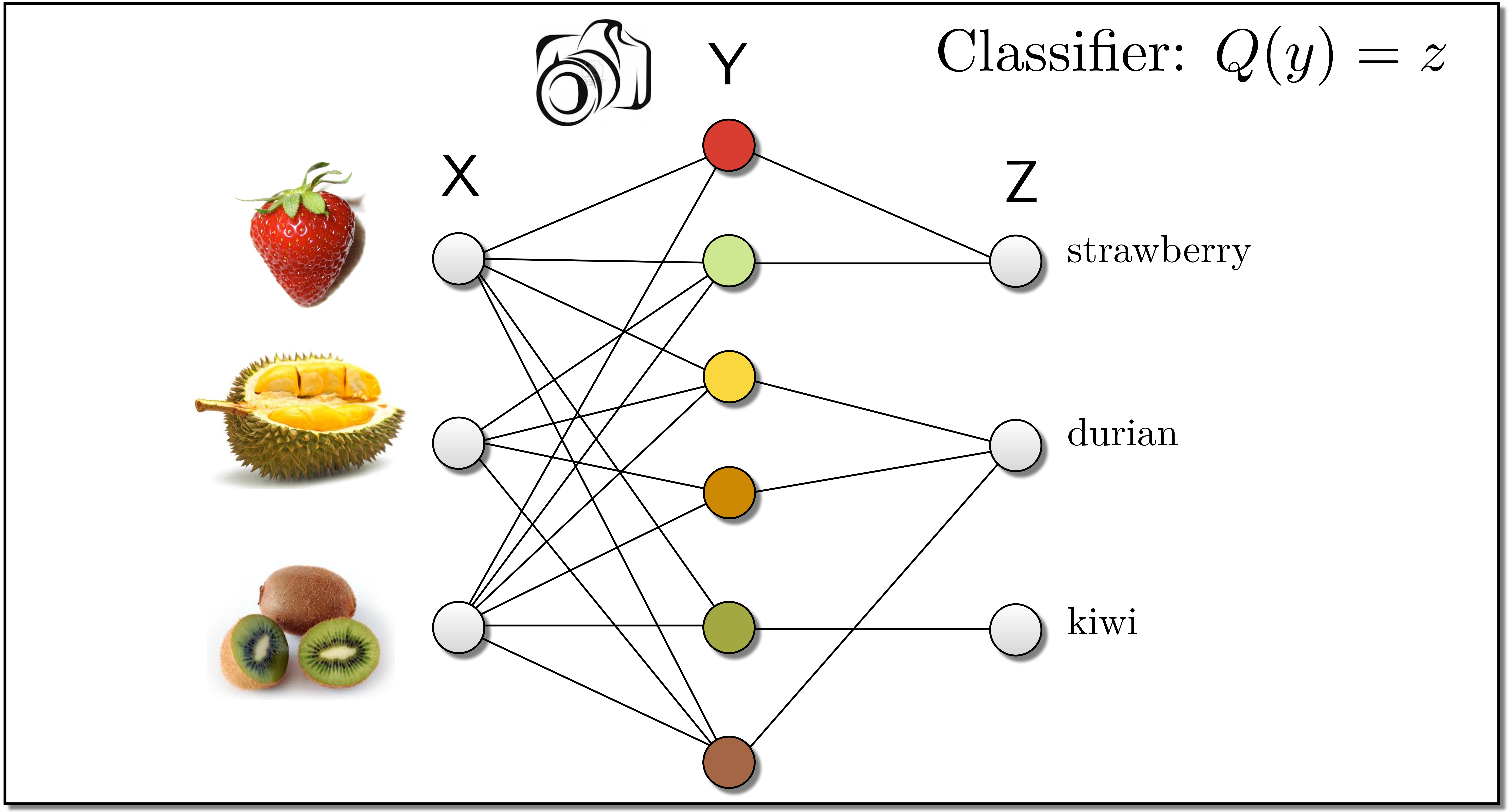


Simple Example of Classification



Assume $\Pr(Y|X)\Pr(X)$ is known

Simple Example of Classification



Machine Learning for Communications

What communications problems can be solved using machine learning?

- Machine learning is a collection of tools
- There are various problems to solve within the field of communications

"Machine learning for communications" means finding the right tool for your problem.

Should I use deep neural networks for my problem?

- If you have a large amount of data, then yes
- No data? Better to use other machine learning techniques

Machine learning is much broader than deep neural networks

Outline

Present three motivating problems and solutions using machine learning:

1. Low-latency communications: Soft-input decoding of BCH codes

   Solution: Deep neural network as a decoding algorithm

2. Optimal quantization of channels

   Solution: K-means algorithm for quantization

3. Fixed-point implementation of LDPC decoders targeted at VLSI

   Solutions: Application of K-means algorithm to novel max-LUT method

Reliable Communications over Unreliable Channels

Good channel (few errors) — high code rate R (few parity bits)

Reliable Communications over Unreliable Channels

Good channel (few errors) — high code rate R (few parity bits)

Bad channel (many errors) — low code rate R (many parity bits)

Reasons for Success of LDPC Codes

Low-density parity-check (LDPC) codes are widely used. In communications standards:

- 5G
- WiFi, WiMax, video broadcasting
- Ethernet over twisted pair
- Flash memories, SSD drives, hard drives

Reasons for success of LDPC code:

- LDPC codes are good codes — as block length increases, can approach Shannon limit
- LDPC decoding complexity is linear in the block length

Low-Density Parity-Check (LDPC) Codes

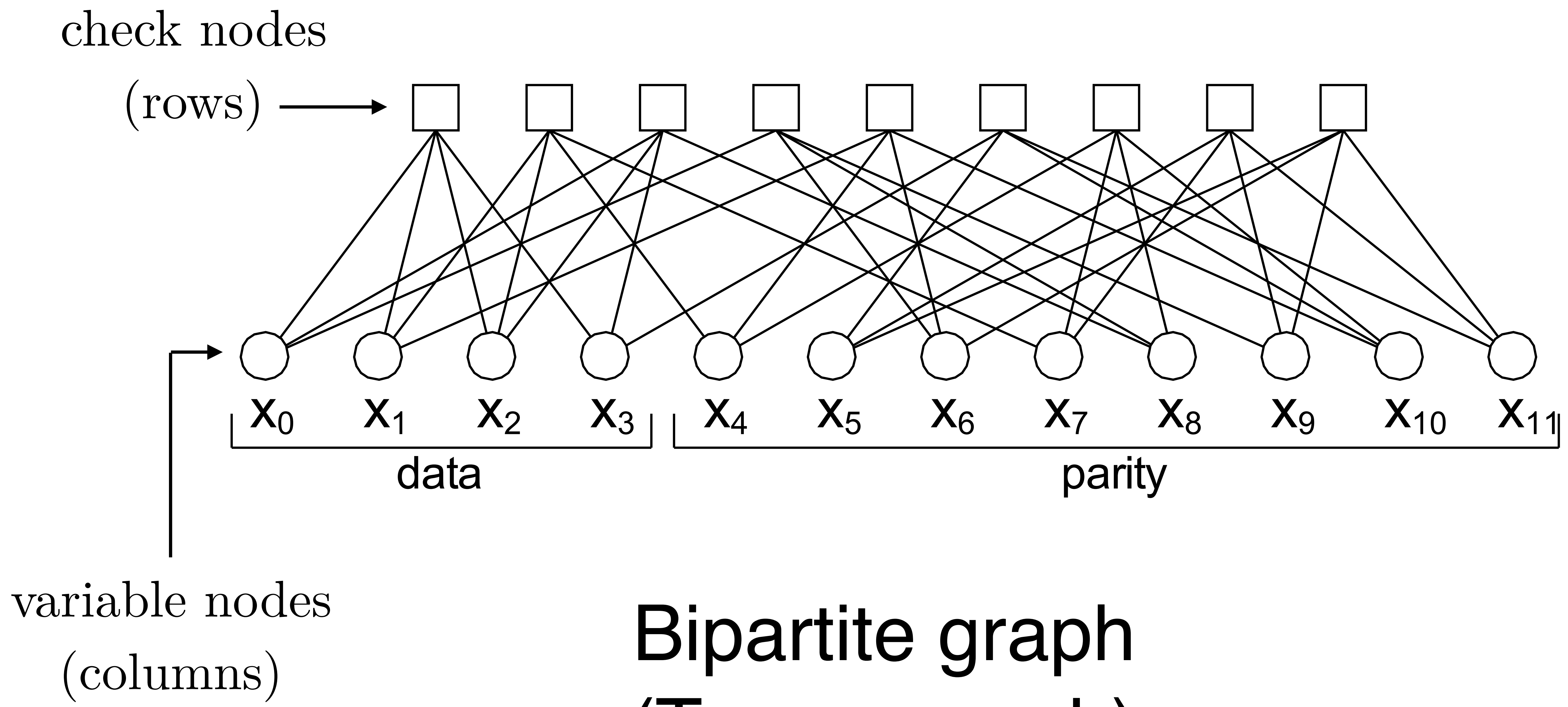
LDPC code is defined by a low-density parity-check matrix H

A codeword $\mathbf{x}$ satisfies $H\mathbf{x} = 0 \bmod 2$

$H =$

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
1	1	1	1	0	0	0	0	0	0	0	0
0	1	1	0	1	0	0	1	0	0	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	1	1	0	0
0	1	0	0	1	0	1	0	0	0	1	0
0	0	0	1	0	1	0	0	0	0	1	1
0	0	0	0	1	0	0	1	1	0	1	0
0	0	0	0	0	1	0	1	0	1	0	1
0	0	0	0	0	1	1	0	0	1	0	1

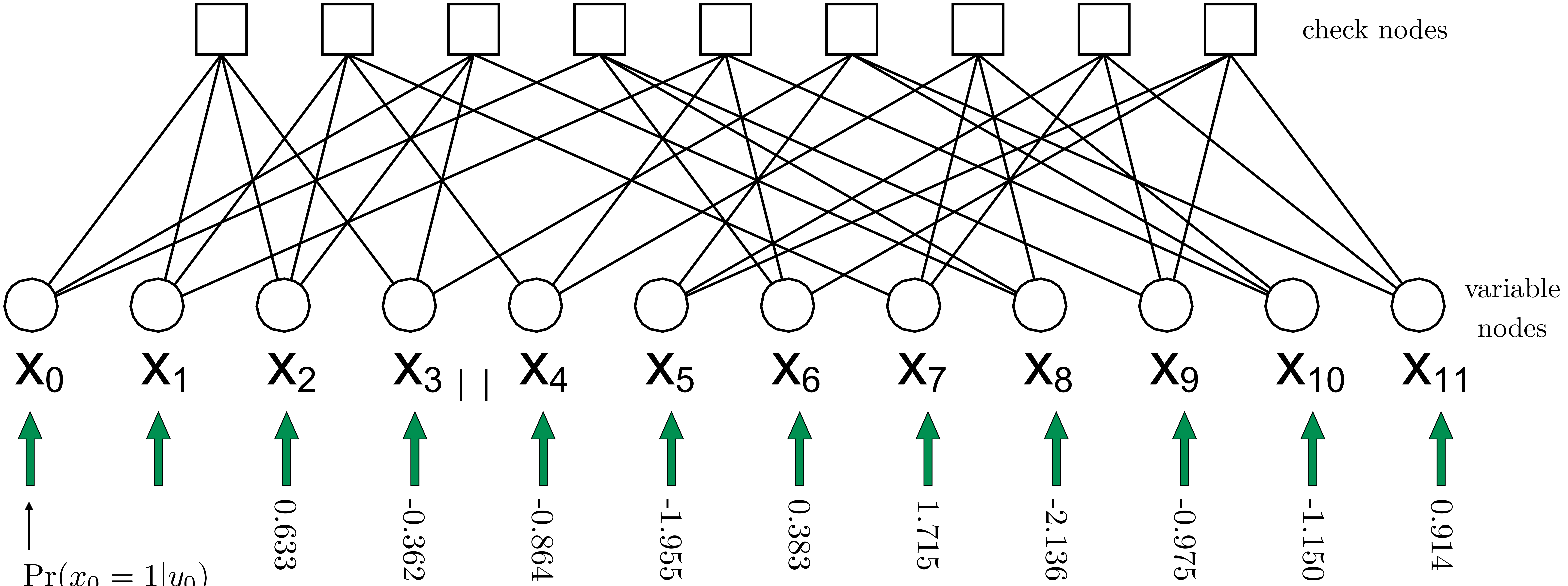
Parity-Check Matrix



Bipartite graph
(Tanner graph)

Recall decoder attempts to solve: $\max_{\mathbf{x}\in\mathcal{C}} \Pr(\mathbf{y}|\mathbf{x})$

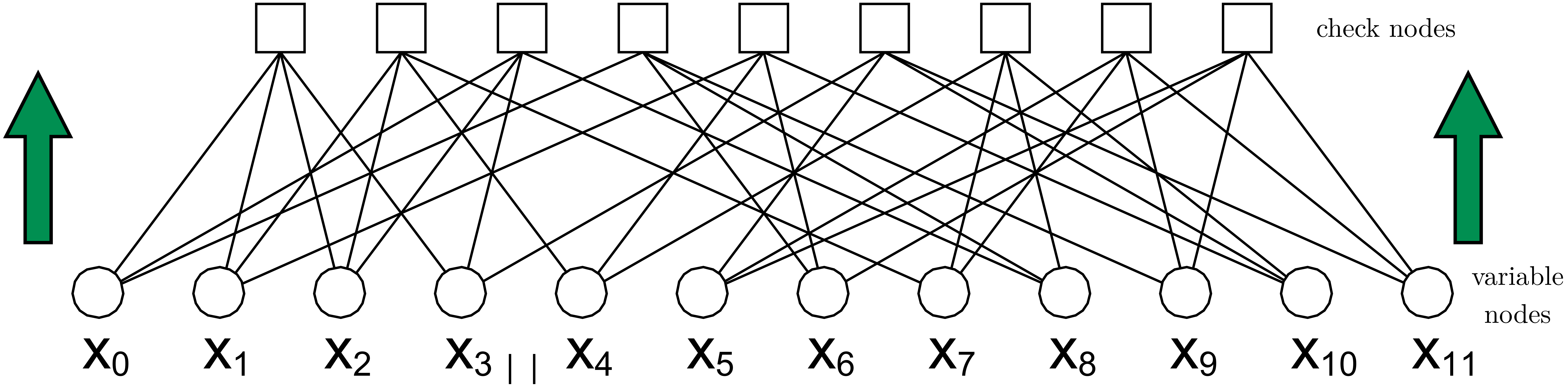
Decoding LDPC Codes
Input from channel



$$\log \frac{\Pr(x_0 = 1|y_0)}{\Pr(x_0 = 0|y_0)} = -0.4512$$

input from channel

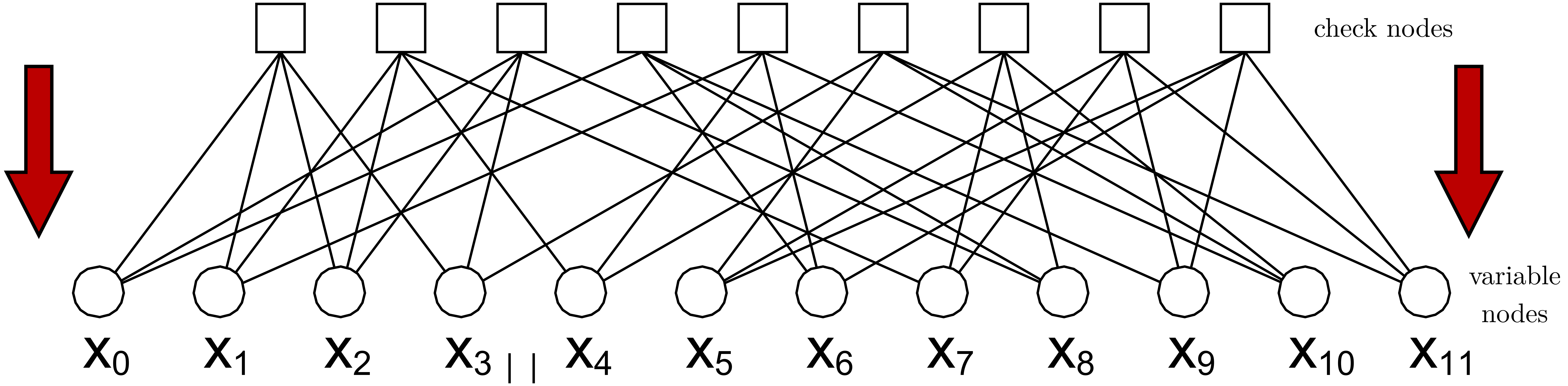
Decoding LDPC Codes
Variable-to-check messages



Iteration 1 (first half):

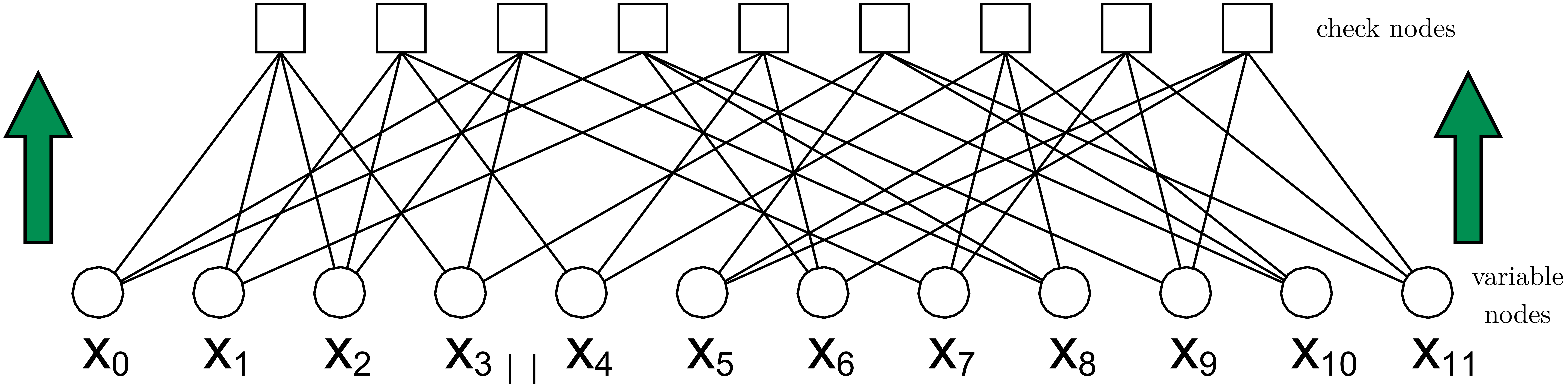
pass channel messages to check nodes

Decoding LDPC Codes
Check-to-Variable Messages



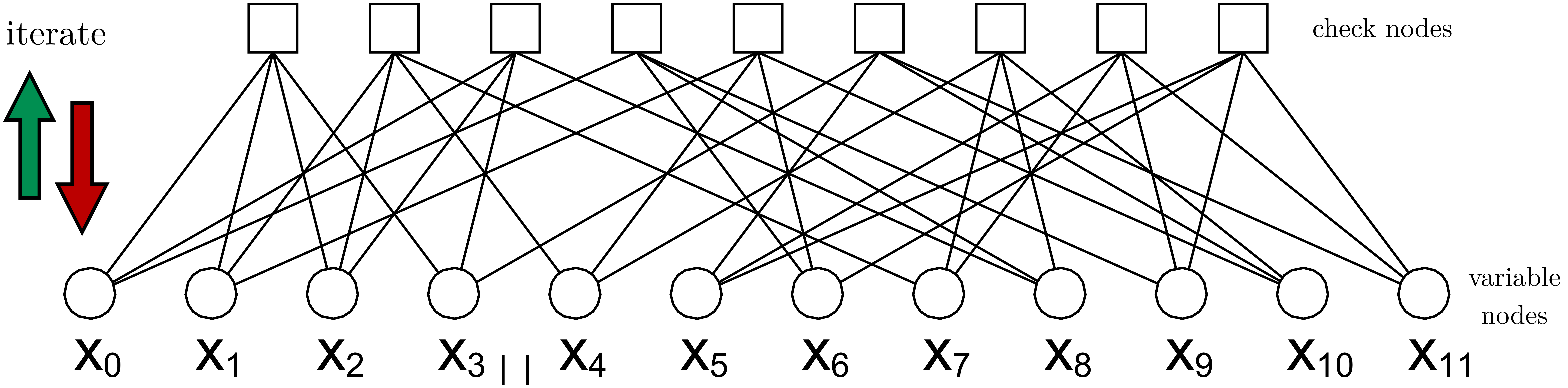
Iteration 1 (second half):
check nodes perform processing,
pass results to variable nodes

Decoding LDPC Codes
Variable-to-check messages



Iteration 2 (first half):
variable nodes perform processing,
pass results to check nodes

Decoding LDPC Codes Continue Iteratively



In practice, perform 5 to 50 iterations. Stop when:

- Codeword is detected $H\mathbf{x} = 0$
- maximum number of iterations reached

Motivation 1: Low-Latency Communications

Ultra-low latency communications is a key component of 5G wireless networks

Enable IoT-like applications: Real-time control in autonomous vehicles, factory automation, robots, UAVs and more.

For highly reliable physical layer, short block-length error-correcting codes are needed.

BCH codes are error-correcting codes with an algebraic construction

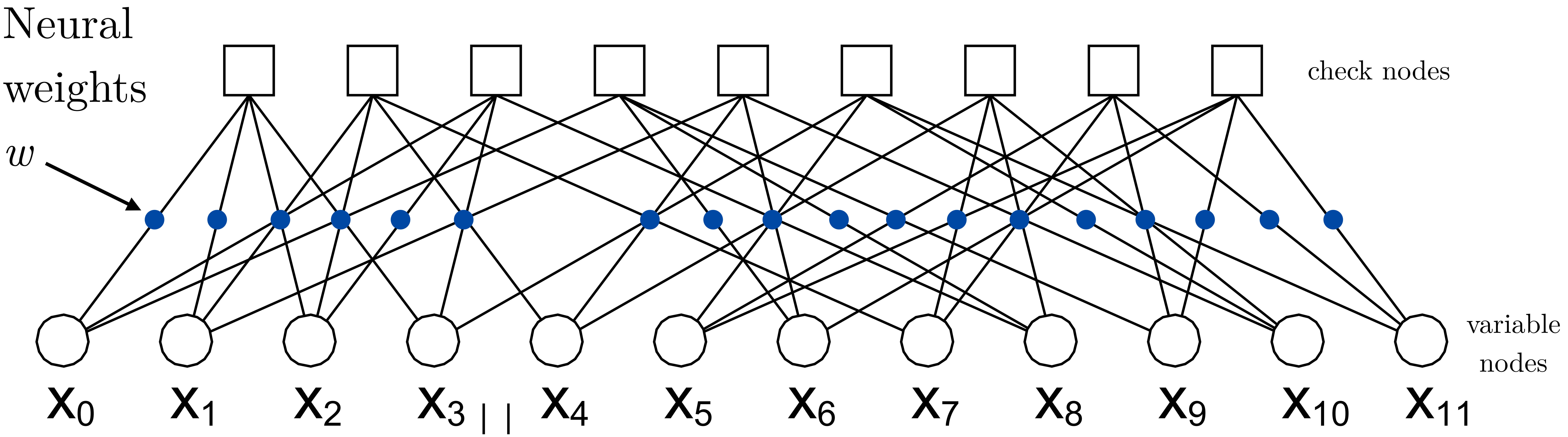
At short-to-medium block lengths, they are the best known codes

Decoding BCH Codes Using Belief Propagation?

- There are good hard-input decoding algorithms for BCH codes
- There are no good soft-input decoding algorithms
  - ✳ BCH codes have **high-density** check matrix
  - ✳ Traditional belief-propagation decoding does not work well because the graph is dense, not sparse.
- Can we make a good soft-input BCH decoder using a deep neural network?

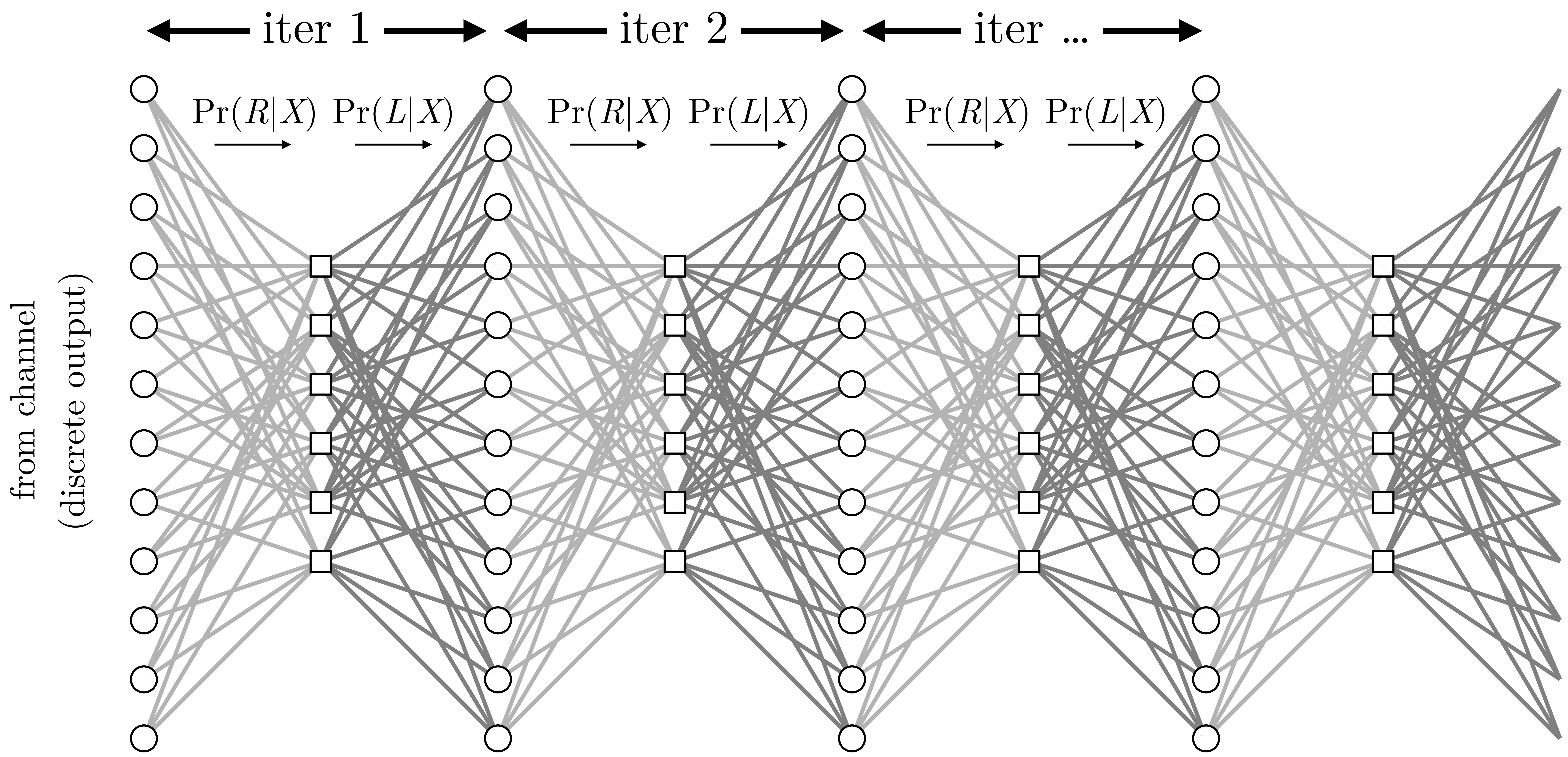
Adding Neural Weights to BP Decoder

- BCH codes also have a graph, but it is dense



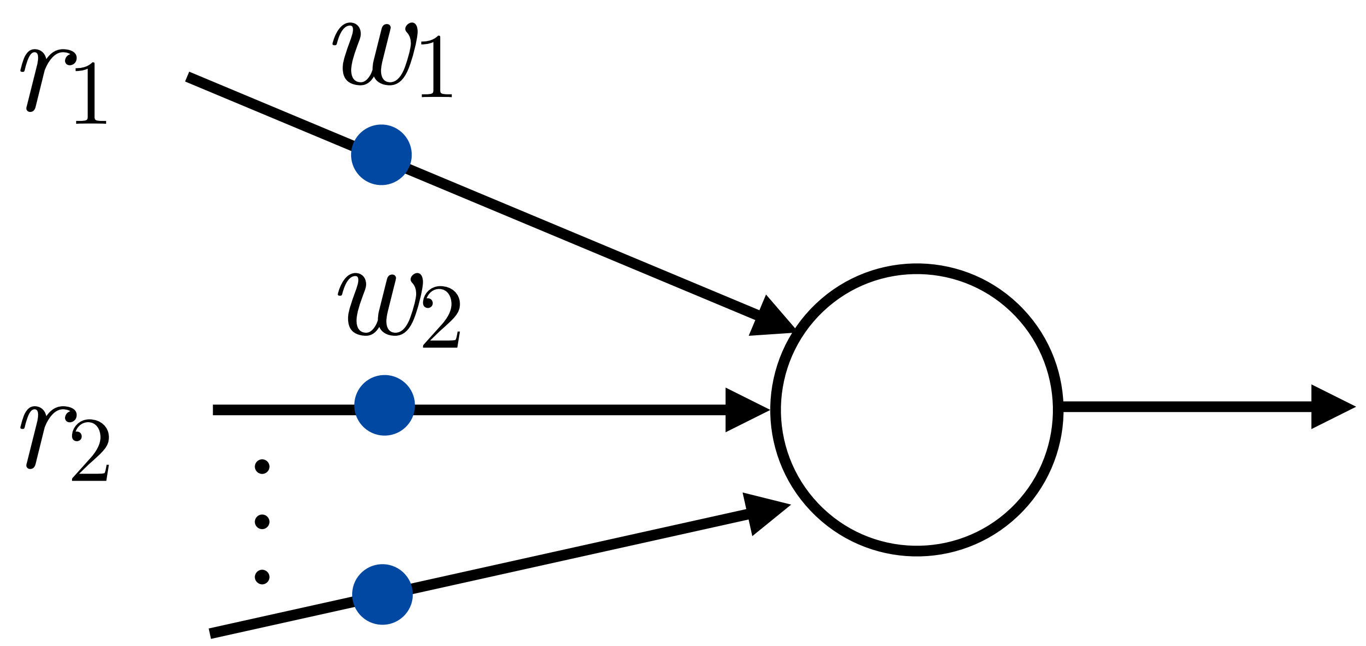
- Cycles in the graph degrade performance of algorithm
- Add neural weights to reduce negative effects of short cycles

LDPC Iterations Unwrap the Graph



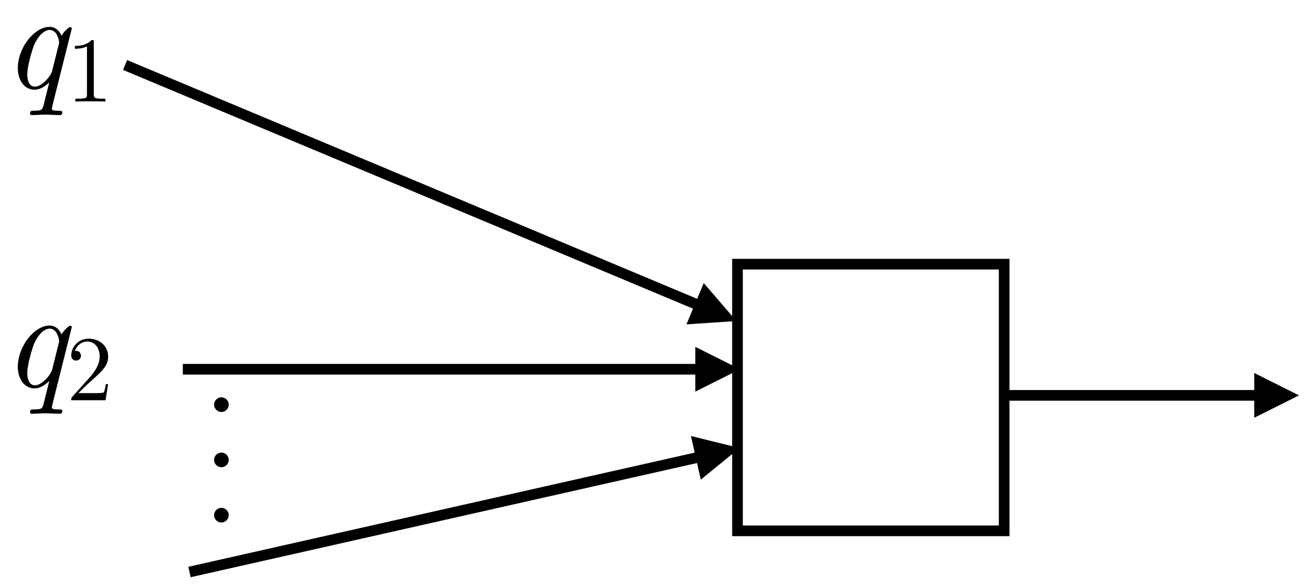
Standard Belief-Propagation Decoding

Variable node



$$q_{v,c} = \tanh\left(\frac{1}{2}\left(l_v + \sum_{i \in \mathcal{N}(v)\backslash c} r_{i,v} \right)\right)$$

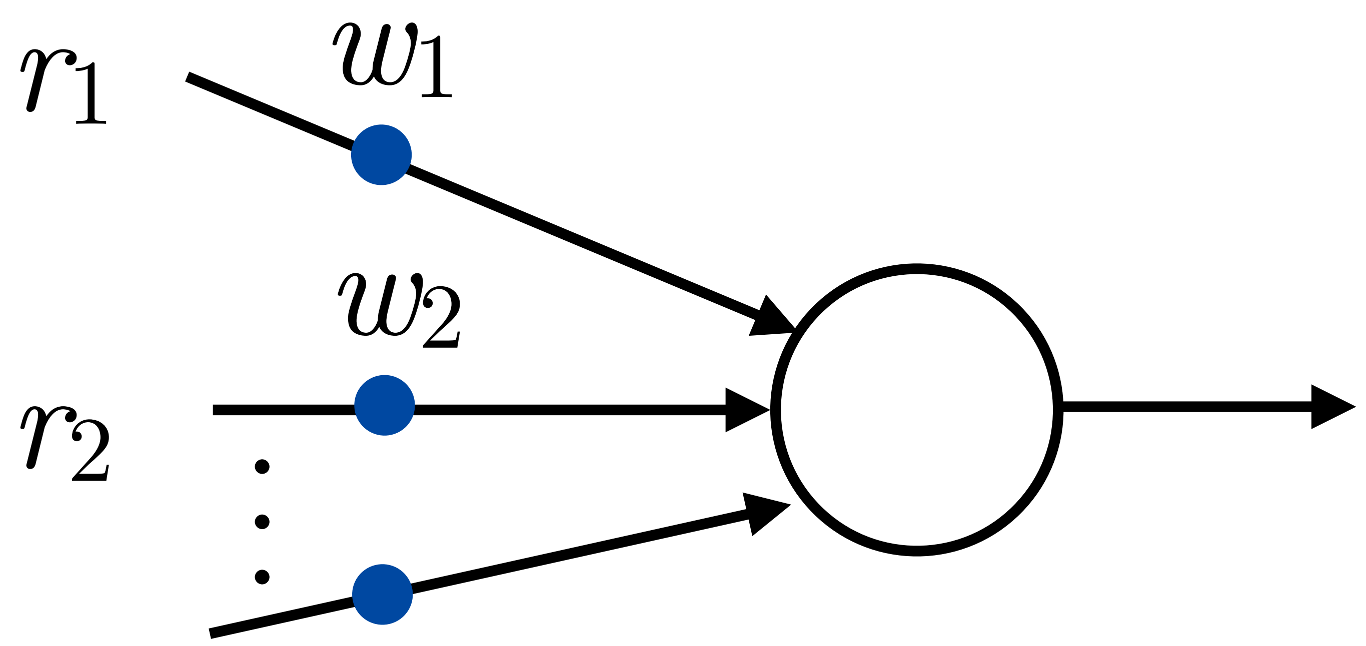
Check node



$$r_{c,v} = 2\tanh^{-1}\left(\prod_{i \in \mathcal{M}(c)\backslash v} q_{v,i} \right)$$

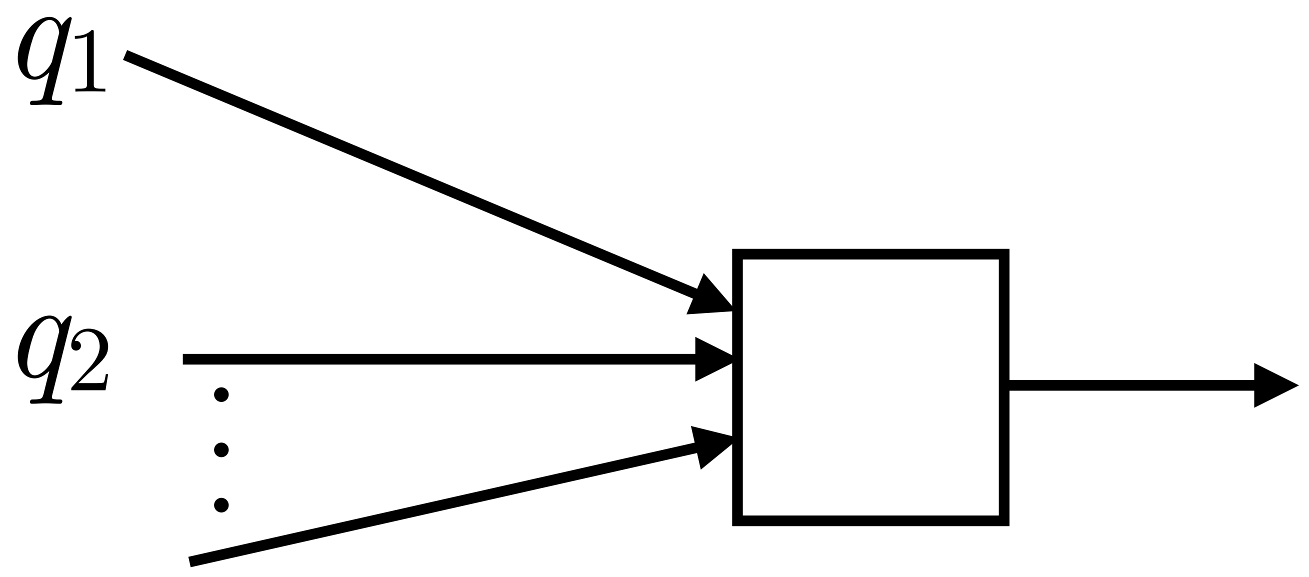
Modify to Add Weights

Variable node



$$q_{v,c} = \tanh\left(\frac{1}{2}\left(w_v l_v + \sum_{i \in \mathcal{N}(v)\backslash c} w_{i,v} r_{i,v}\right)\right)$$

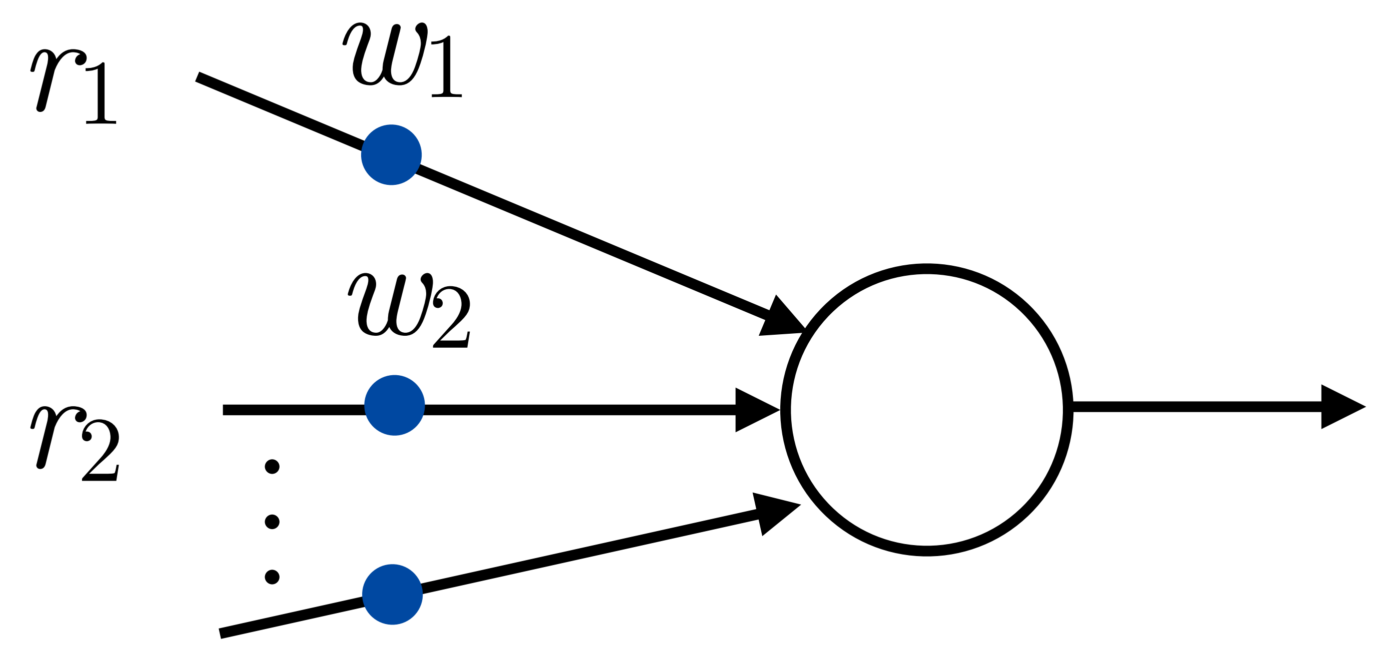
Check node



$$r_{c,v} = 2\tanh^{-1}\left(\prod_{i \in \mathcal{M}(c)\backslash v} q_{v,i}\right)$$

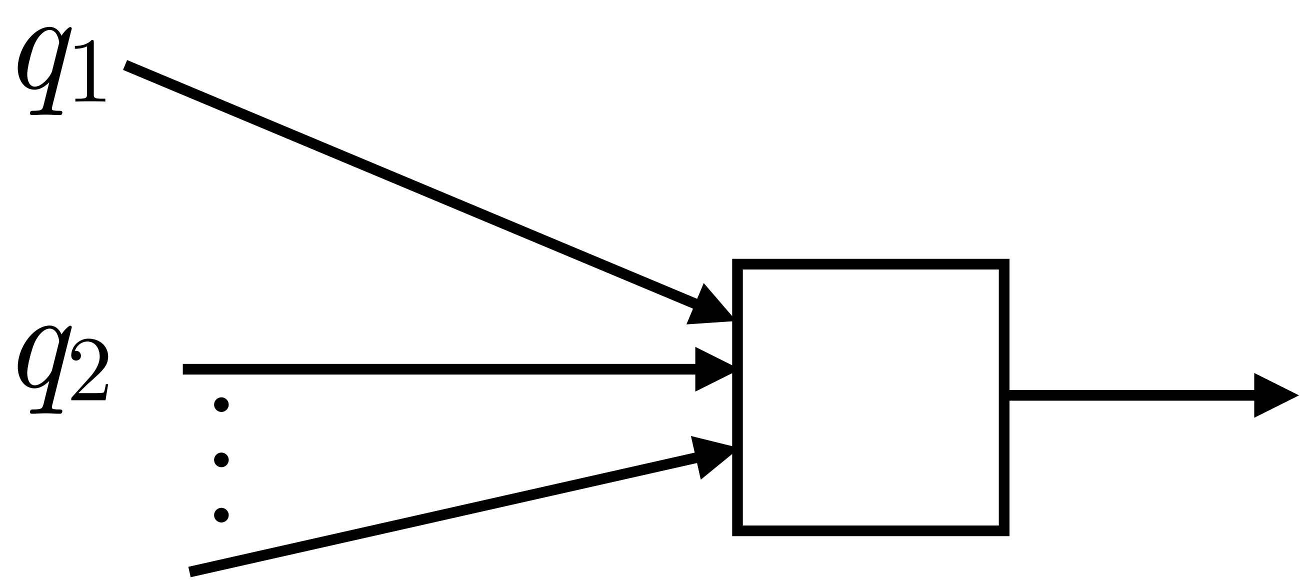
Compare Variable Node & Pereptron

Variable node



$$q_{v,c} = \tanh\left(\frac{1}{2}\left(w_v l_v + \sum_{i \in \mathcal{N}(v)\backslash c} w_{i,v} r_{i,v}\right)\right)$$

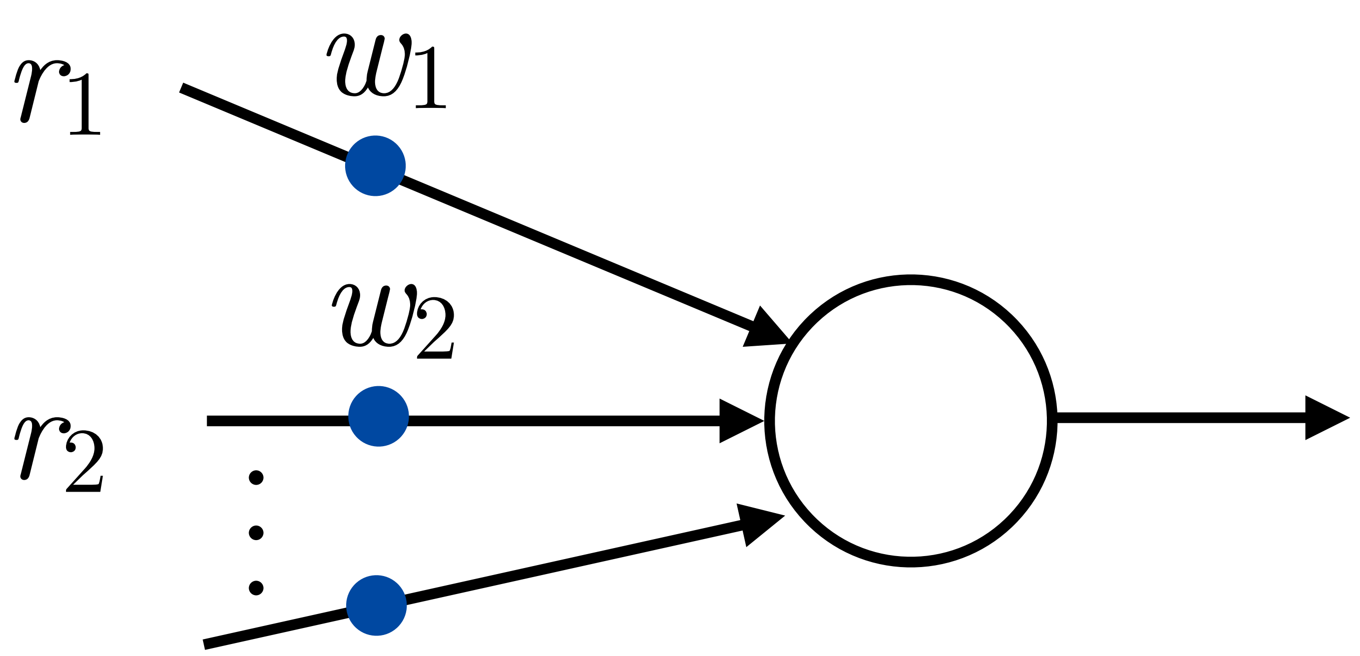
Check node



$$r_{c,v} = 2\tanh^{-1}\left(\prod_{i \in \mathcal{M}(c)\backslash v} q_{v,i}\right)$$

very similar

Perceptron



$$q_{v,c} = \tanh\left(w_0 + \sum_{i=1}^{n} w_i x_i\right)$$

Deep Neural Network: BCH Decoding

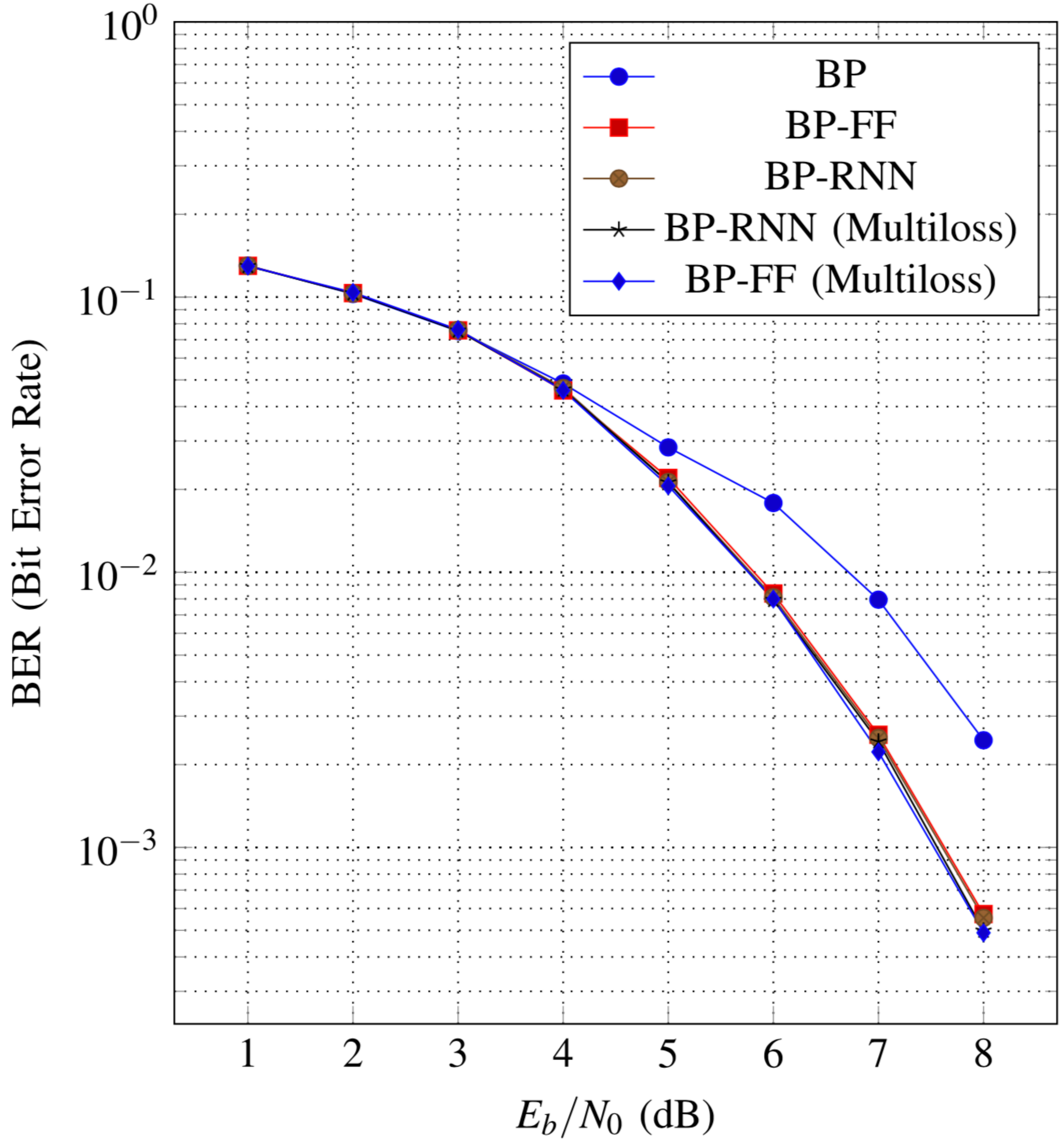
Important insight: train using all-zeros codeword

Training using RMSprop

Multiloss: optimize for all layers, not just the final layer.

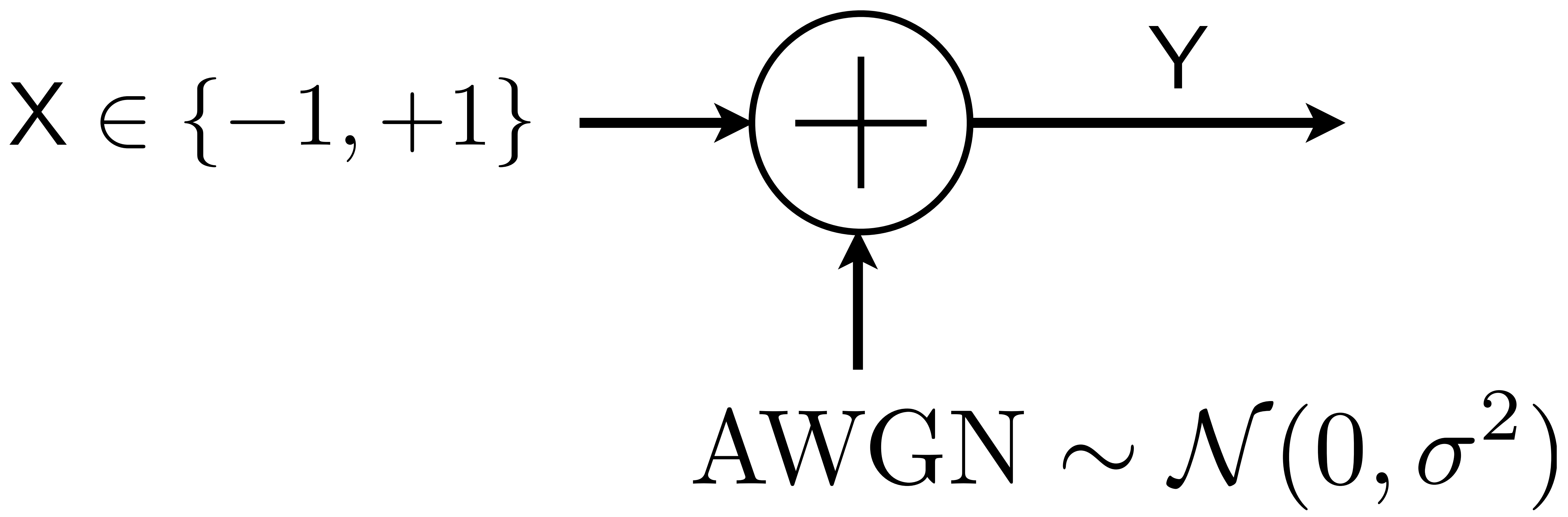
Simulation results show about 1 dB gain over BP. Close to optimal using about 50 iterations.

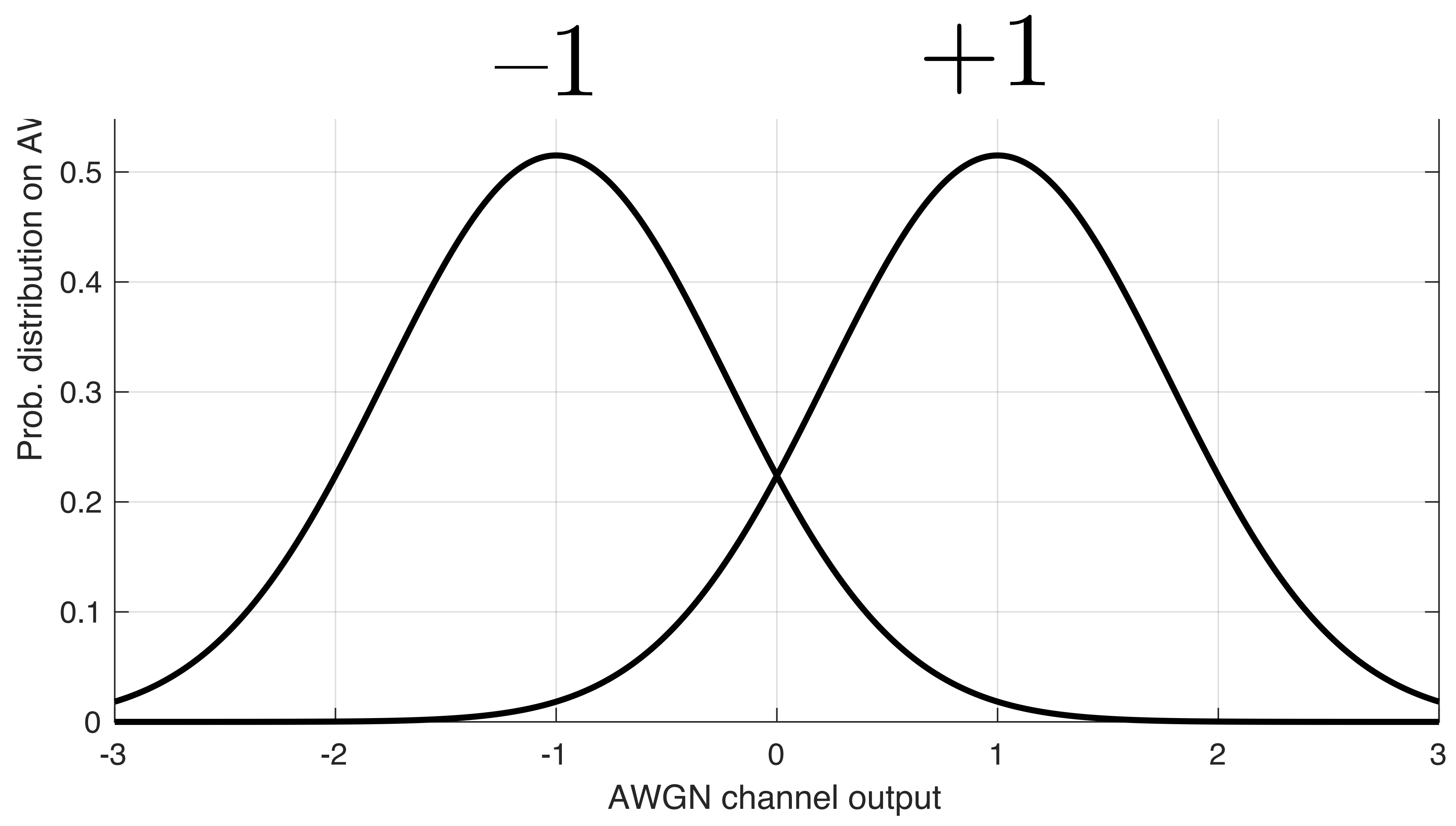
Result: Deep neural network is a practical soft-input decoding of BCH codes



Nachmani, Marciano, Lugosch, Gross, Burshtein, Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE Journal of Selected Topics in Signal Processing*, 12(1), 119-131.

Motivation 2: Optimal Quantization of Channels

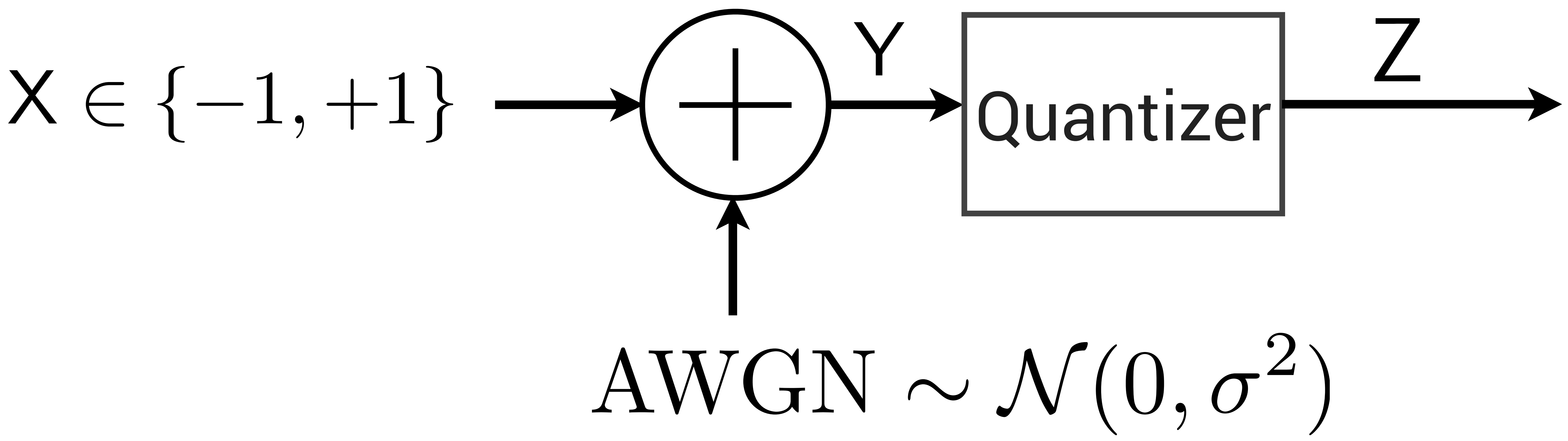


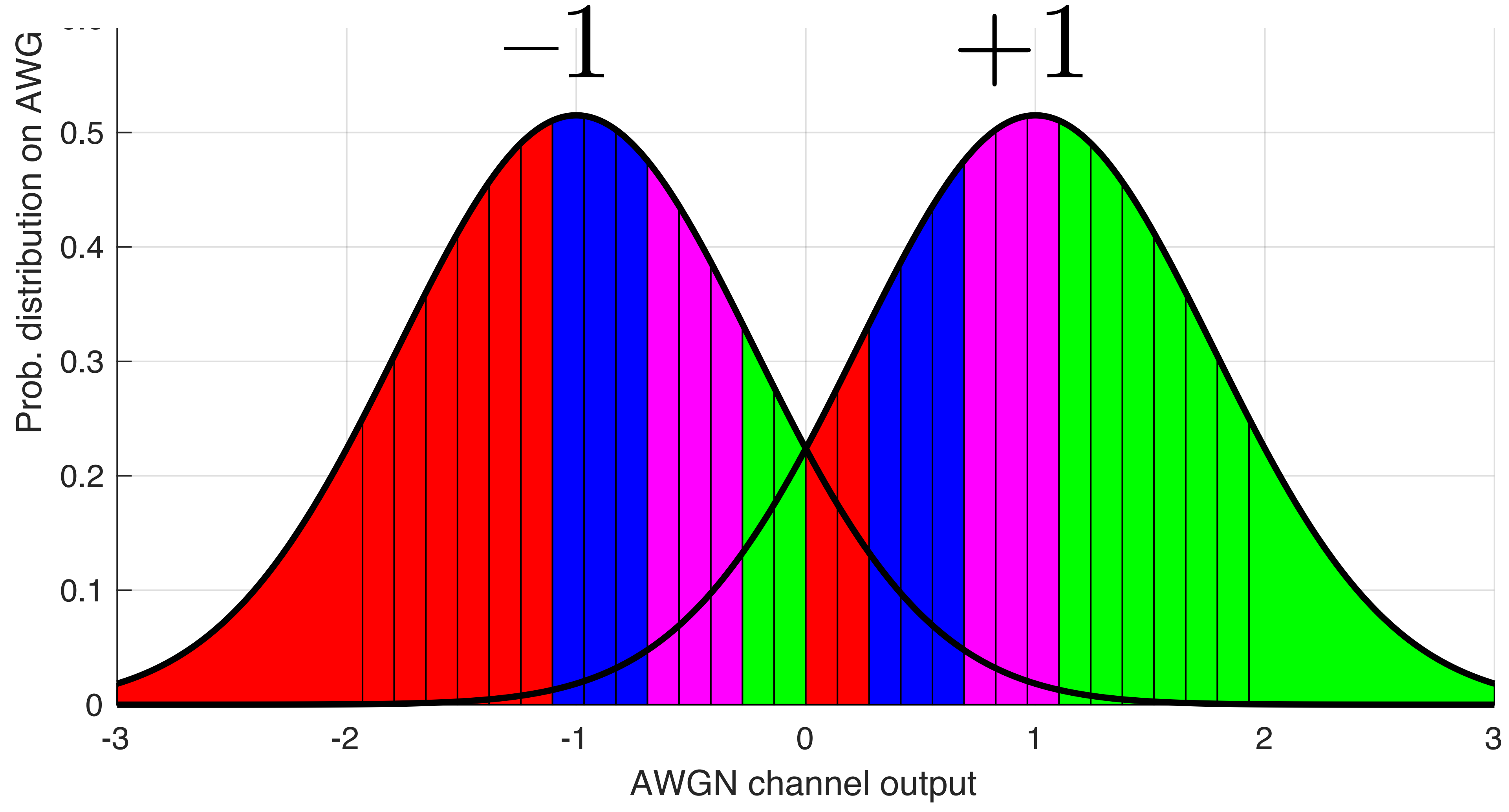


Given a continuous-output channel, we want to create a discrete version

- For example, digital circuits deal with discrete values.

Motivation2 : Optimal Quantization of Channels





Given a continuous-output channel, we want to create a discrete version

- For example, digital circuits deal with discrete values.
- A quantizer Q maps real values Y to discrete values $\mathsf{Z} \in \{1, \ldots, M\}$
- How to choose the "quantization boundaries" to $\max I(\mathsf{X}; \mathsf{Z})$?

X. Ma, X. Zhang, H. Yu, and A. Kavcic, "Optimal quantization for soft-decision decoding revisited," in Proc. Int. Symp. Inform. Theory Appl., Xian, China, Oct. 2002

K-Means Algorithm

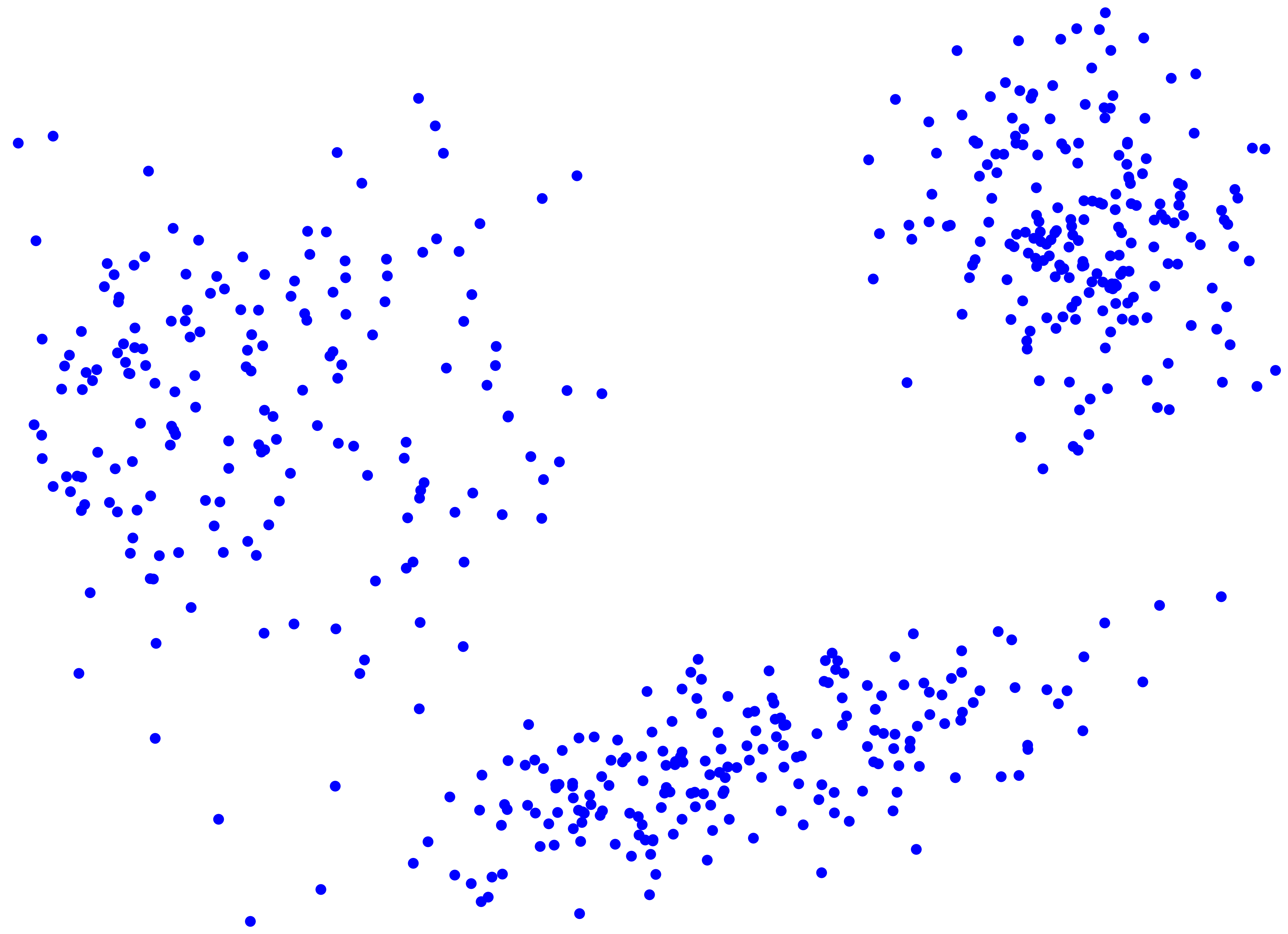
The K-means algorithm is a classification algorithm.

K-means algorithm partitions n observations into k clusters — each observation belongs to the cluster with the nearest mean. Can also be seen as vector quantization.

Attempts to minimize mean-squared of quantization

$$\min_{Q} E\left[(X - Q(X))^2\right]$$

Not optimal, but works well. Widely used in machine learning.



Contents lists available at ScienceDirect

Pattern Recognition Letters

ELSEVIER

journal homepage: www.elsevier.com/locate/patrec

Data clustering: 50 years beyond K-means ☆

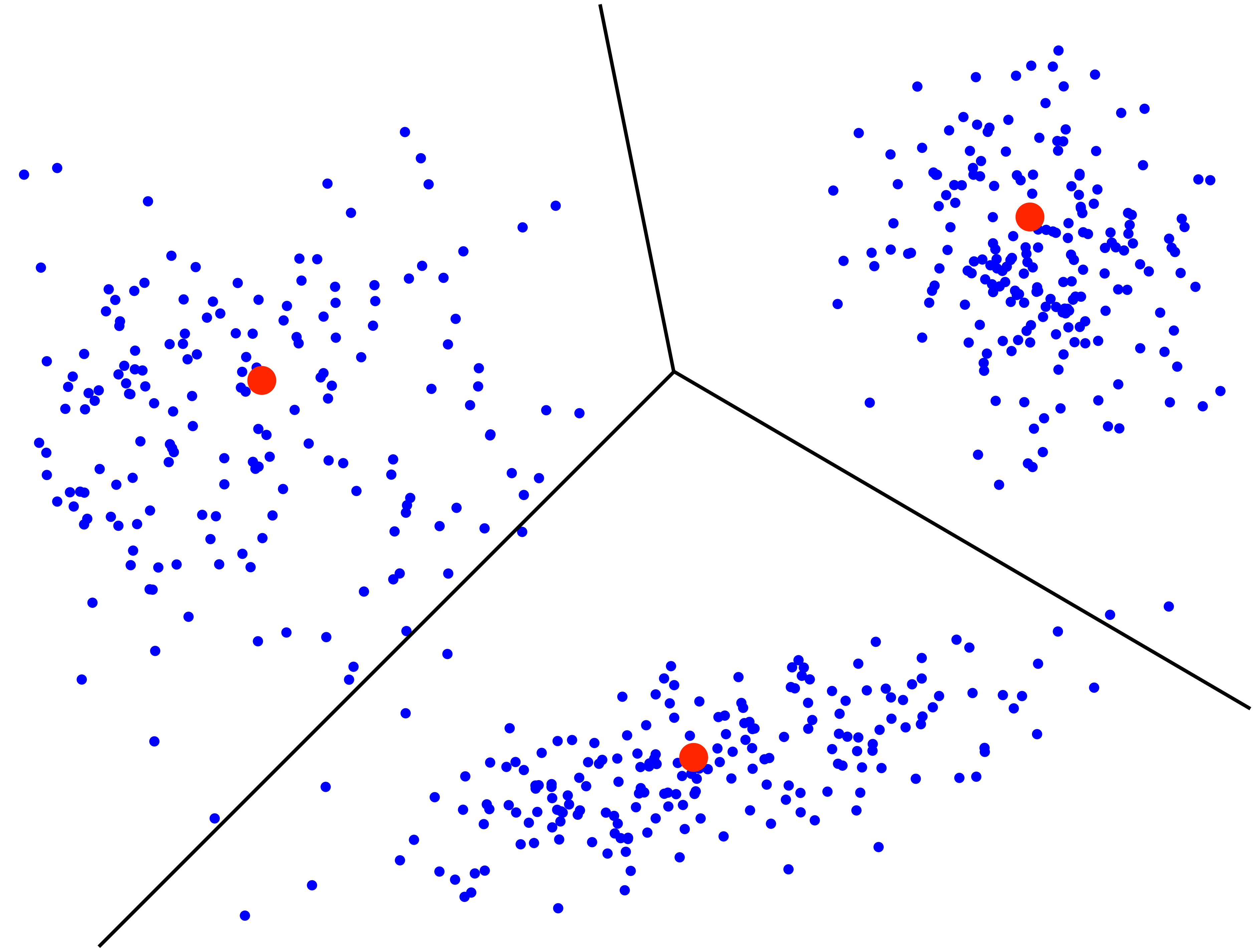
Anil K. Jain *

Department of Computer Science and Engineering, Michigan State University, East Lansing, Michigan 48824, USA
Department of Brain and Cognitive Engineering, Korea University, Anam-dong, Seoul, 136-713, Korea

ARTICLE INFO

ABSTRACT

K-Means Algorithm



The K-means algorithm is a classification algorithm.

K-means algorithm partitions n observations into k clusters — each observation belongs to the cluster with the nearest mean. Can also be seen as vector quantization.

Attempts to minimize mean-squared of quantization

$$\min_{Q} E\left[(X - Q(X))^2\right]$$

Not optimal, but works well. Widely used in machine learning.

Contents lists available at ScienceDirect

Pattern Recognition Letters

ELSEVIER

journal homepage: www.elsevier.com/locate/patrec

Data clustering: 50 years beyond K-means ☆

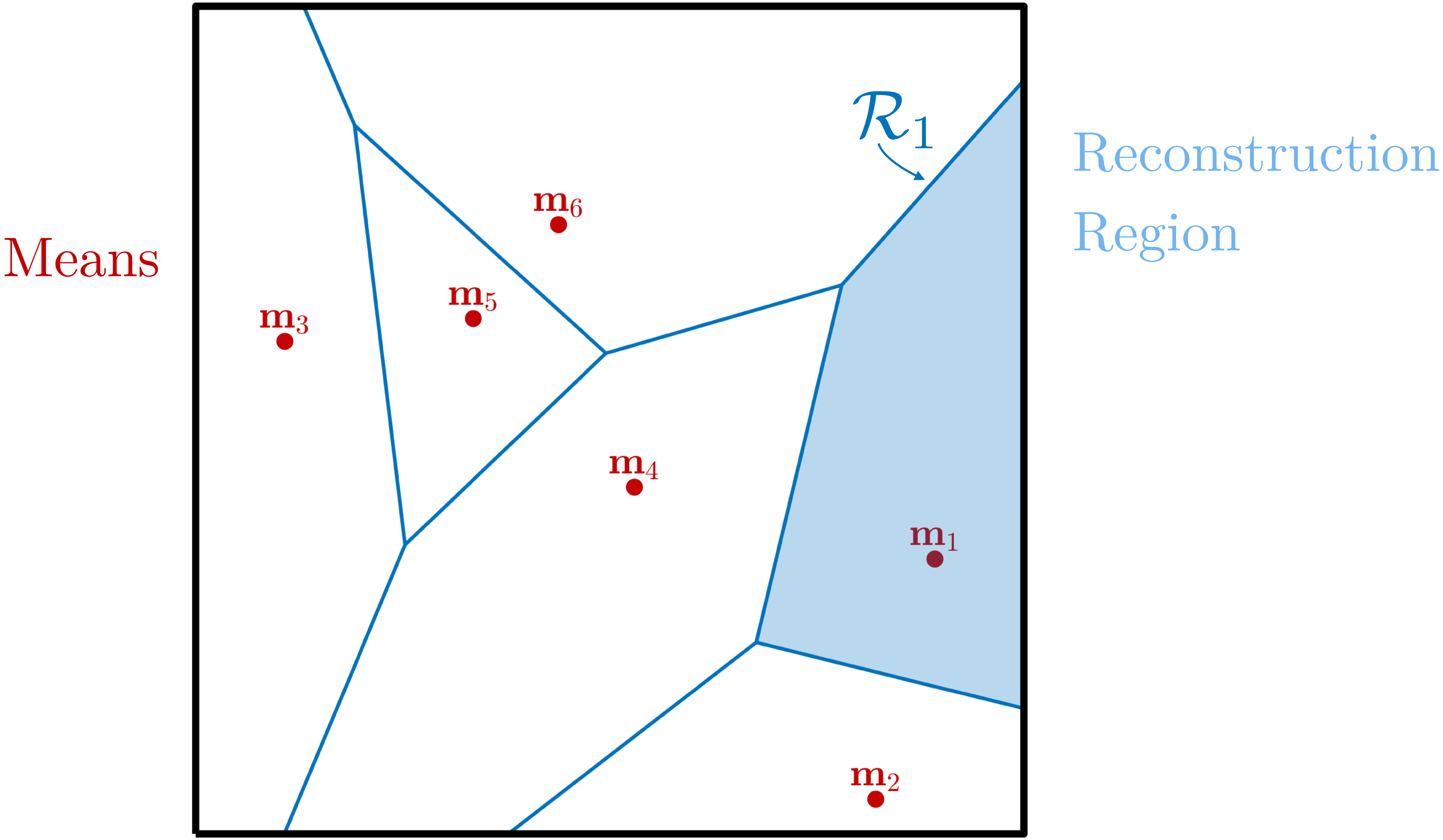
Anil K. Jain *

Department of Computer Science and Engineering, Michigan State University, East Lansing, Michigan 48824, USA
Department of Brain and Cognitive Engineering, Korea University, Anam-dong, Seoul, 136-713, Korea

ARTICLE INFO

ABSTRACT

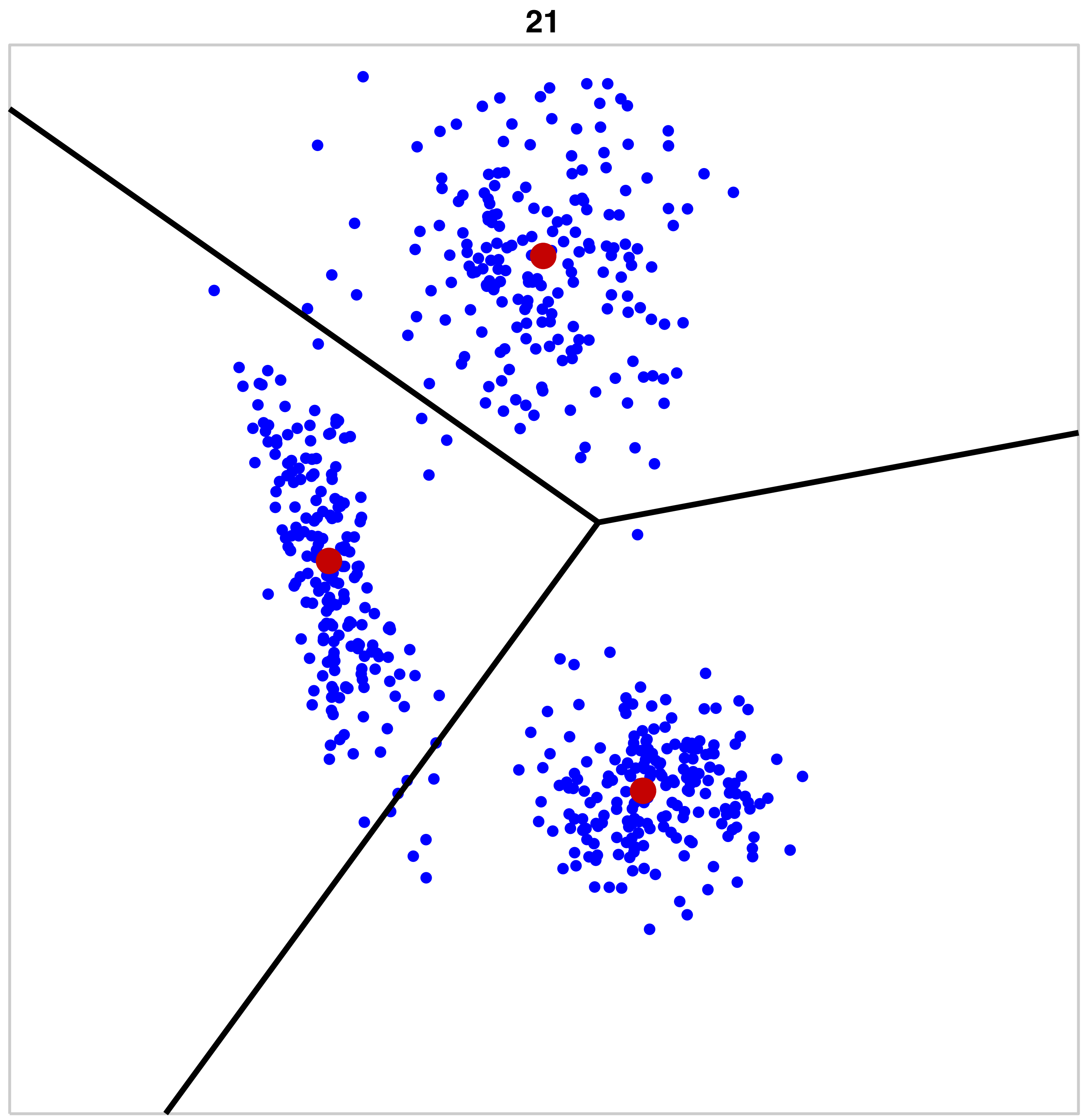
Reconstruction Region and Means



K-Means Algorithm with Euclidean Distance Metric

iterate

1. given n-dimensional data set, randomly choose K means (centroids)
2. **Assignment step** K clusters consists of data points closest to its mean in Euclidean distance
3. **Update step** move the mean to the center of the cluster

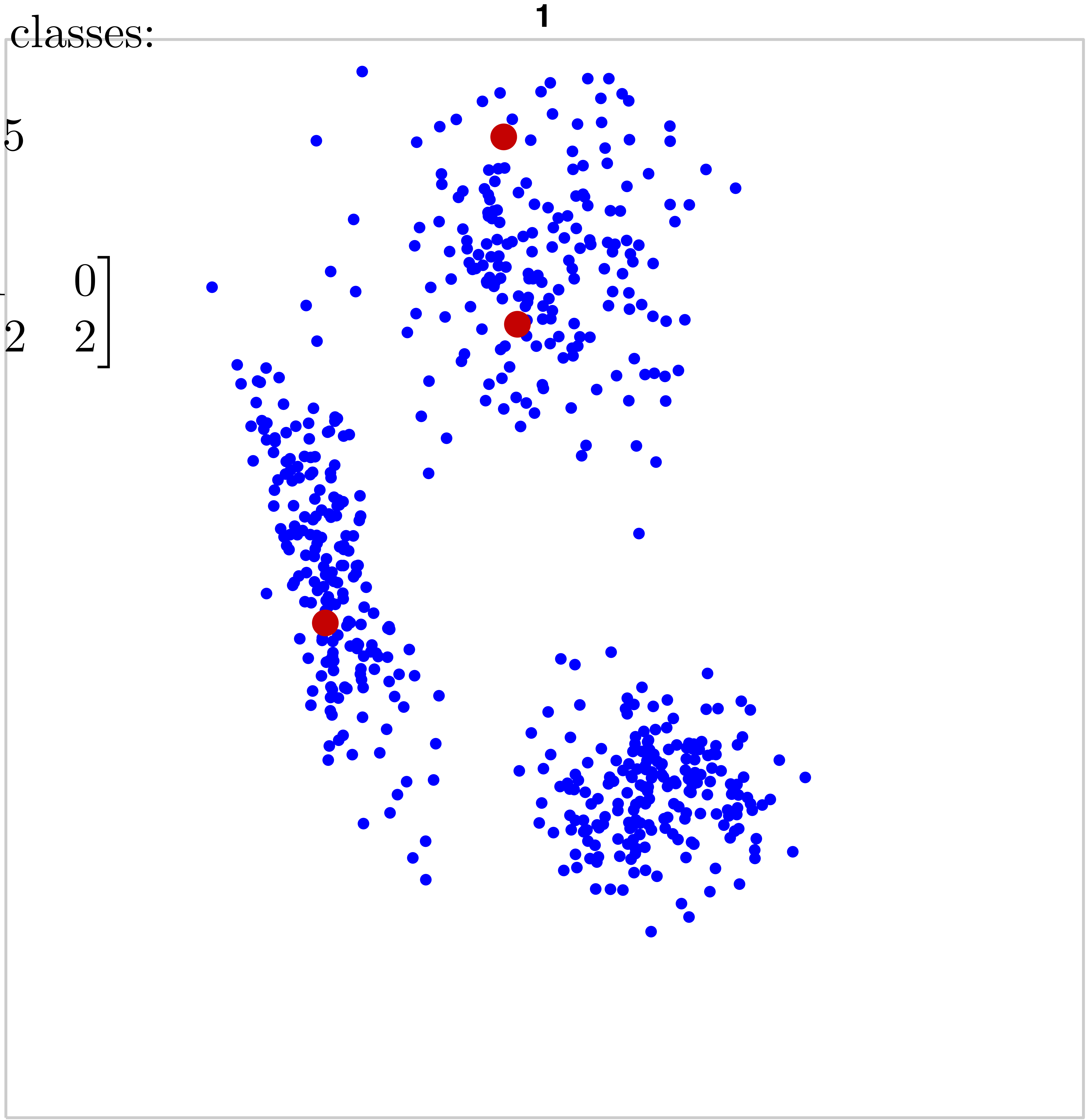


Randomly generate data from one of three classes:

$\mathcal{N}(m_1, v_1)$ where $m_1 = [1, -2], v_1 = 0.25$

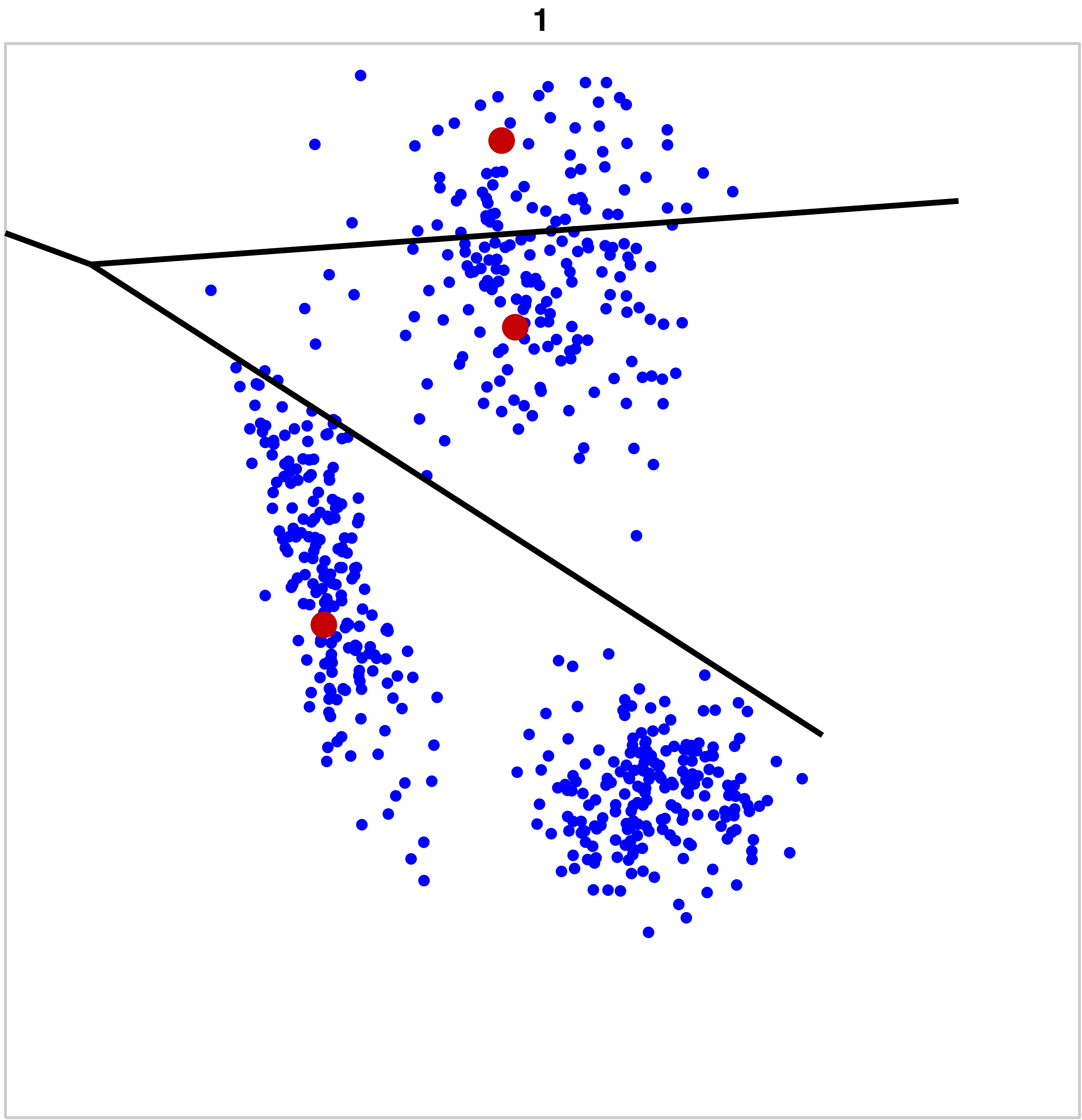
$\mathcal{N}(m_2, v_2)$ where $m_2 = [0, 3], v_2 = 0.5$

$\mathcal{N}(m_3, v_3)$ where $m_3 = [-2, 0], v_3 = \begin{bmatrix} 1 & 0 \\ -2 & 2 \end{bmatrix}$

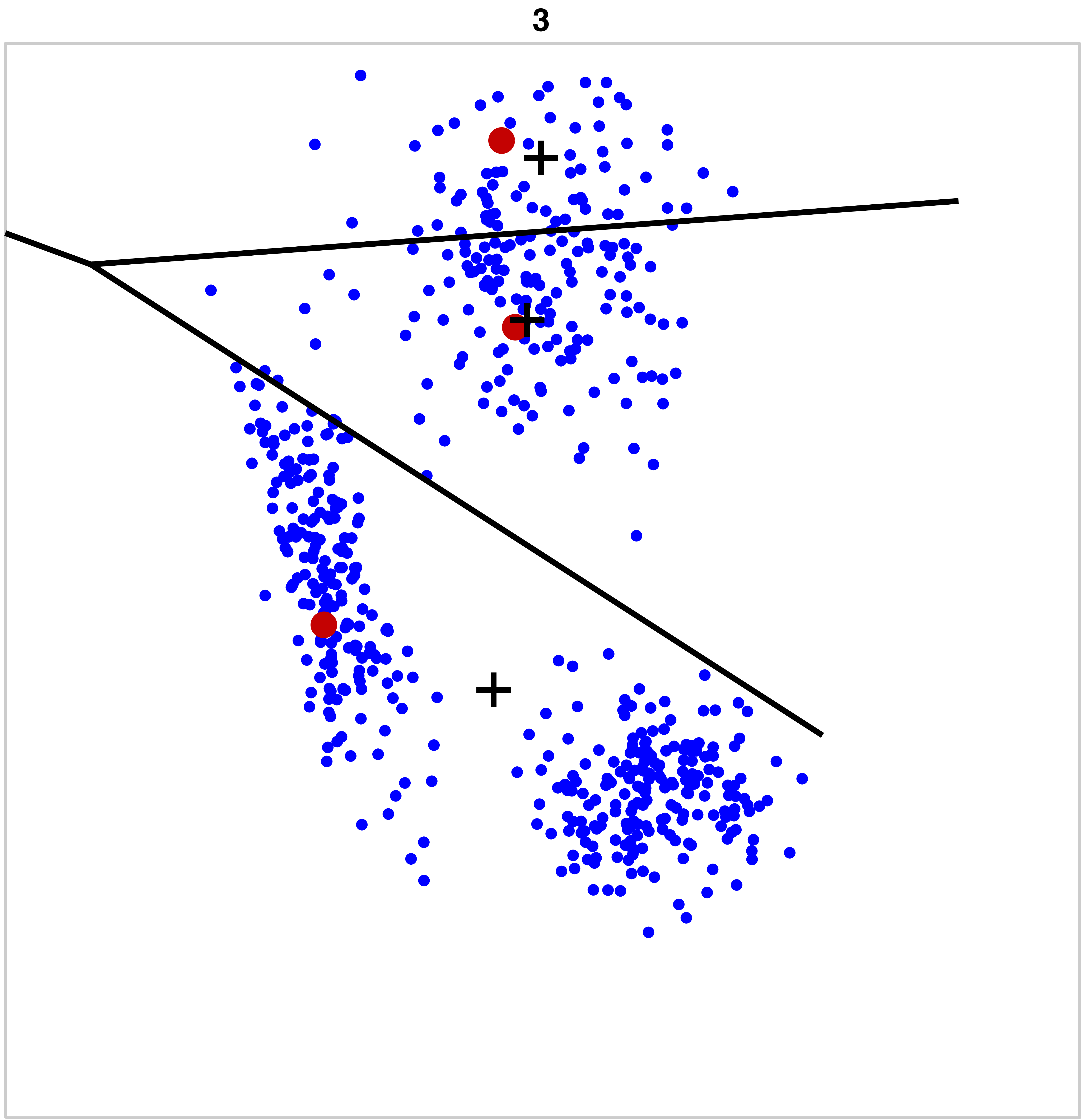


$K = 3$ Initial means are selected randomly

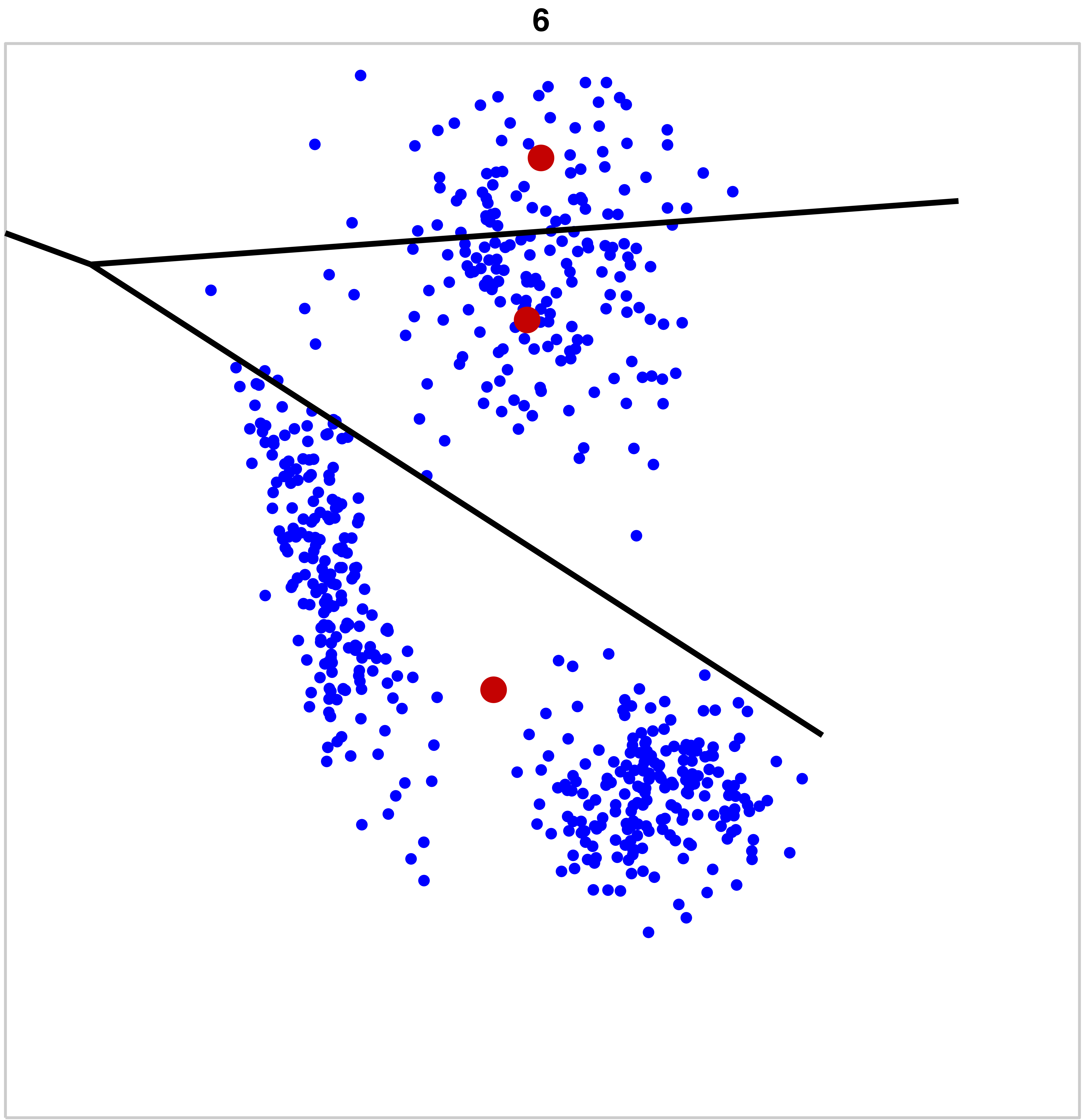
Find reconstruction regions



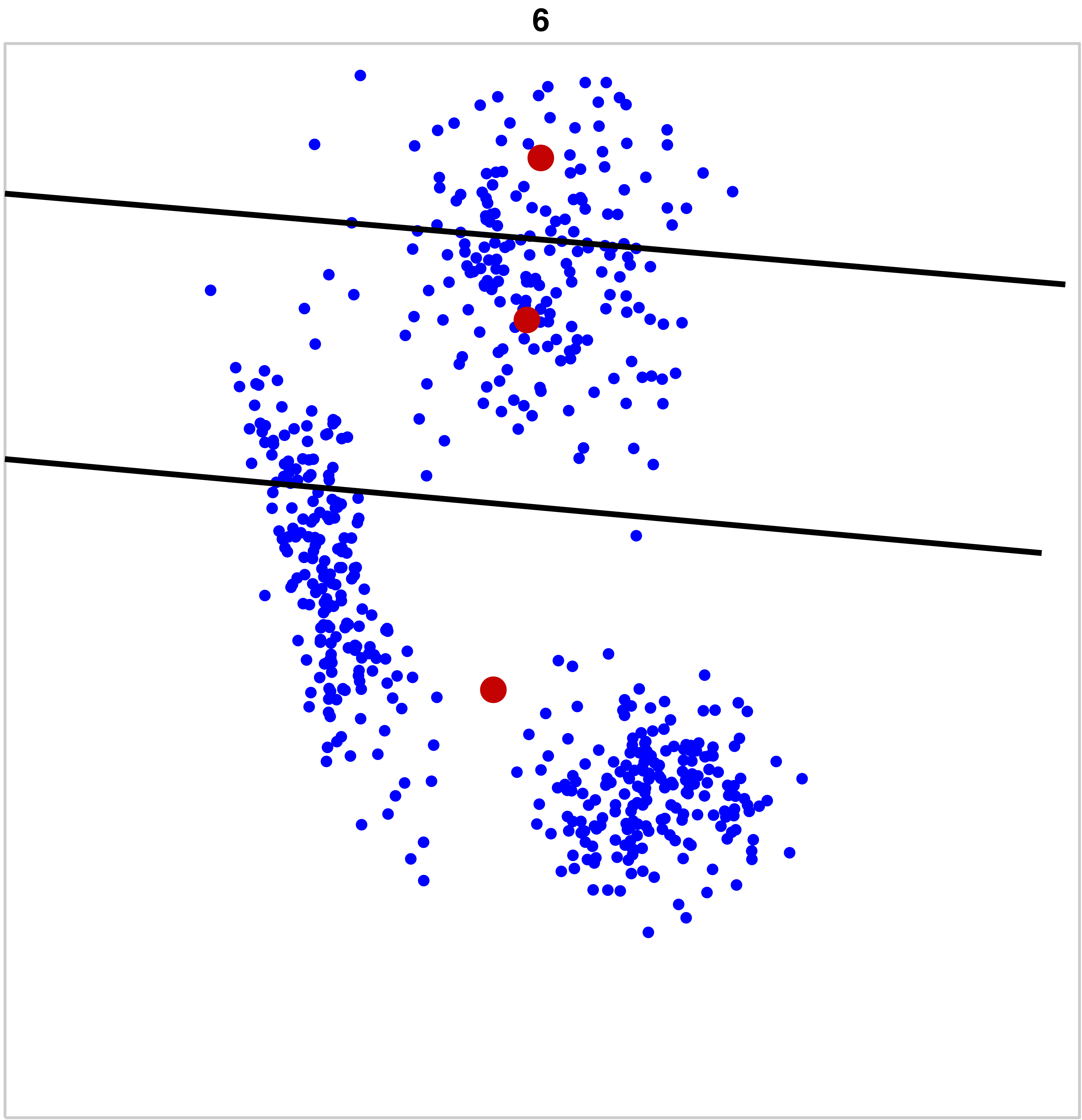
Find center of each cluster



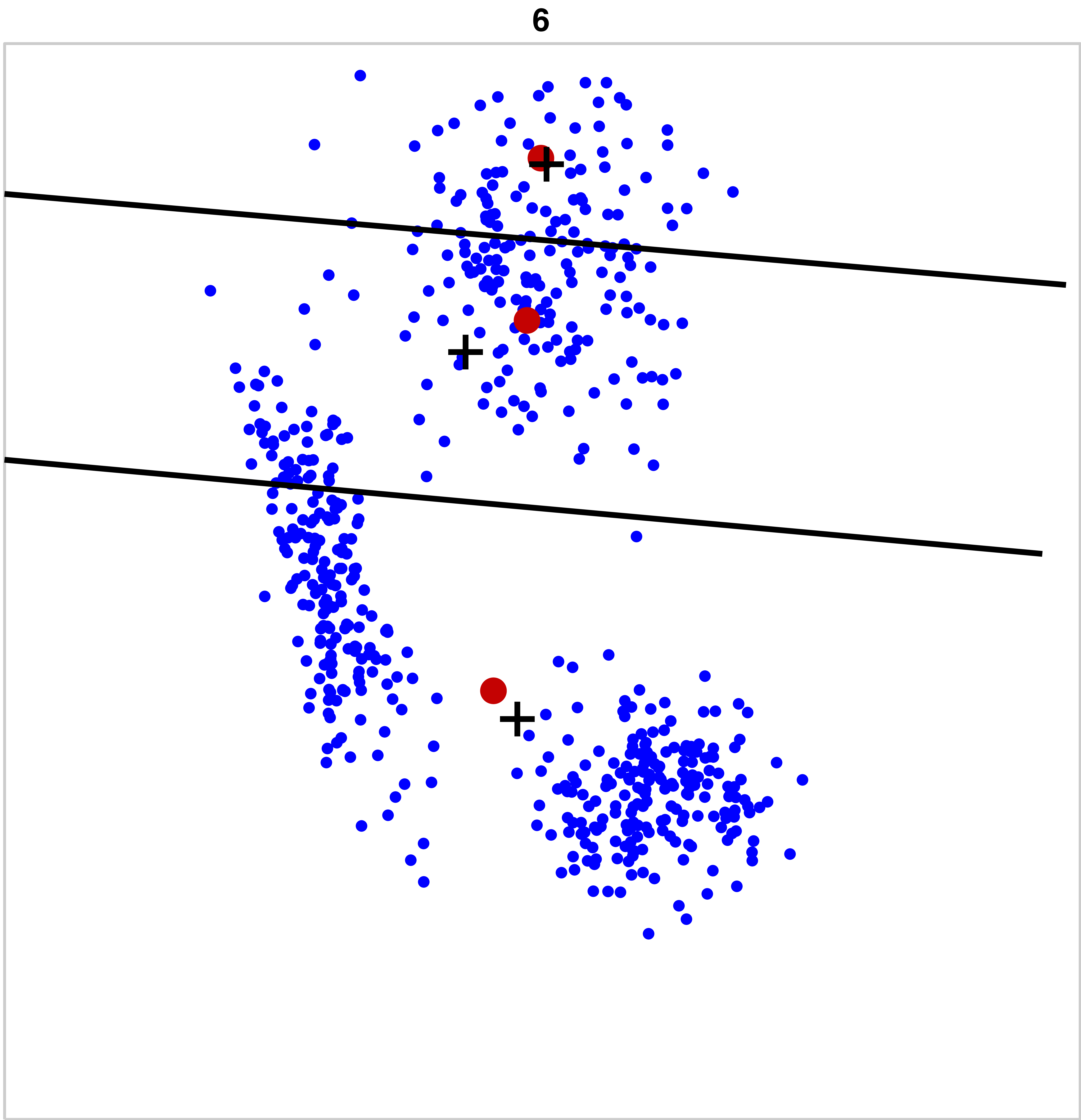
Update the mean



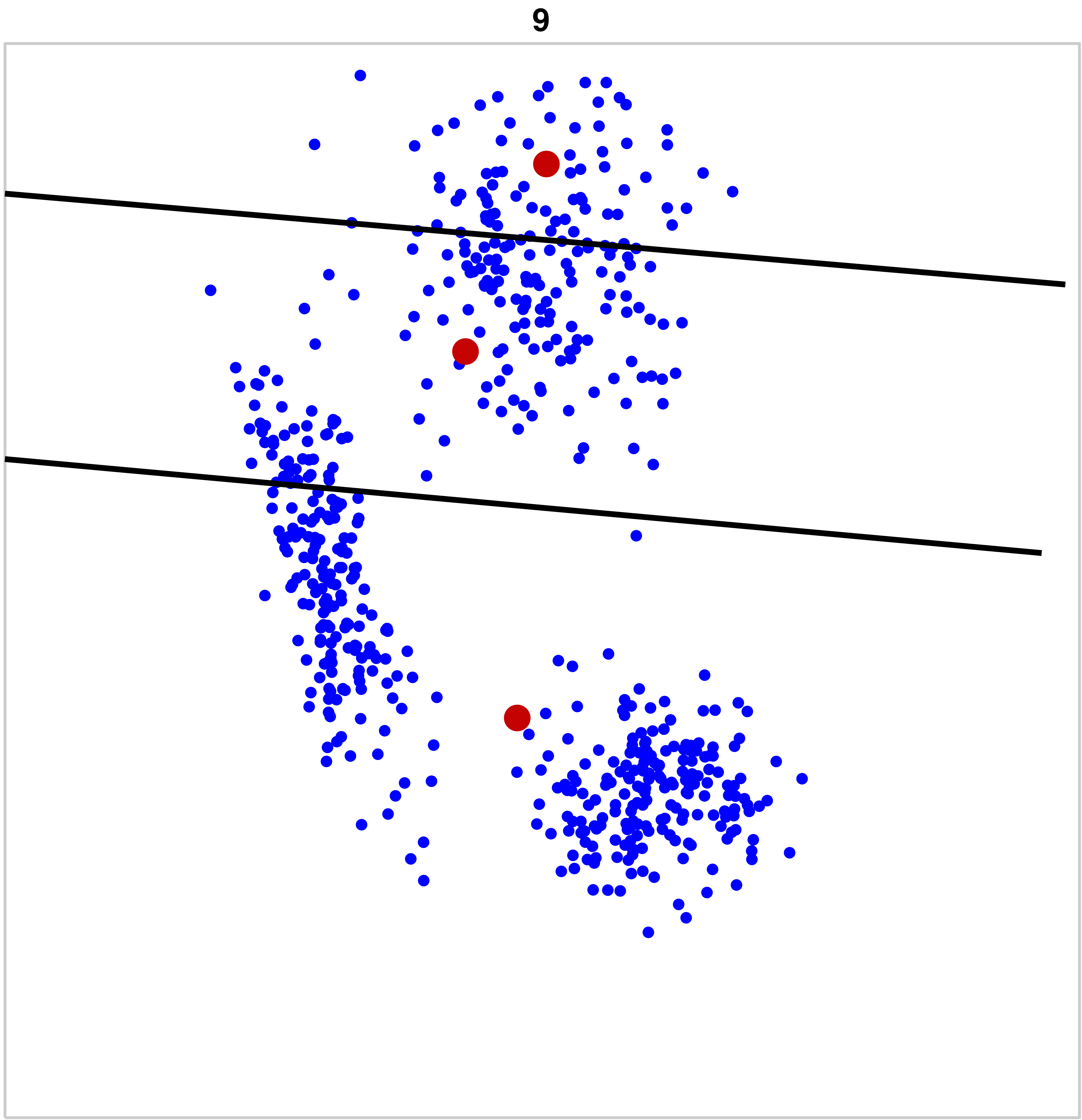
Find reconstruction regions



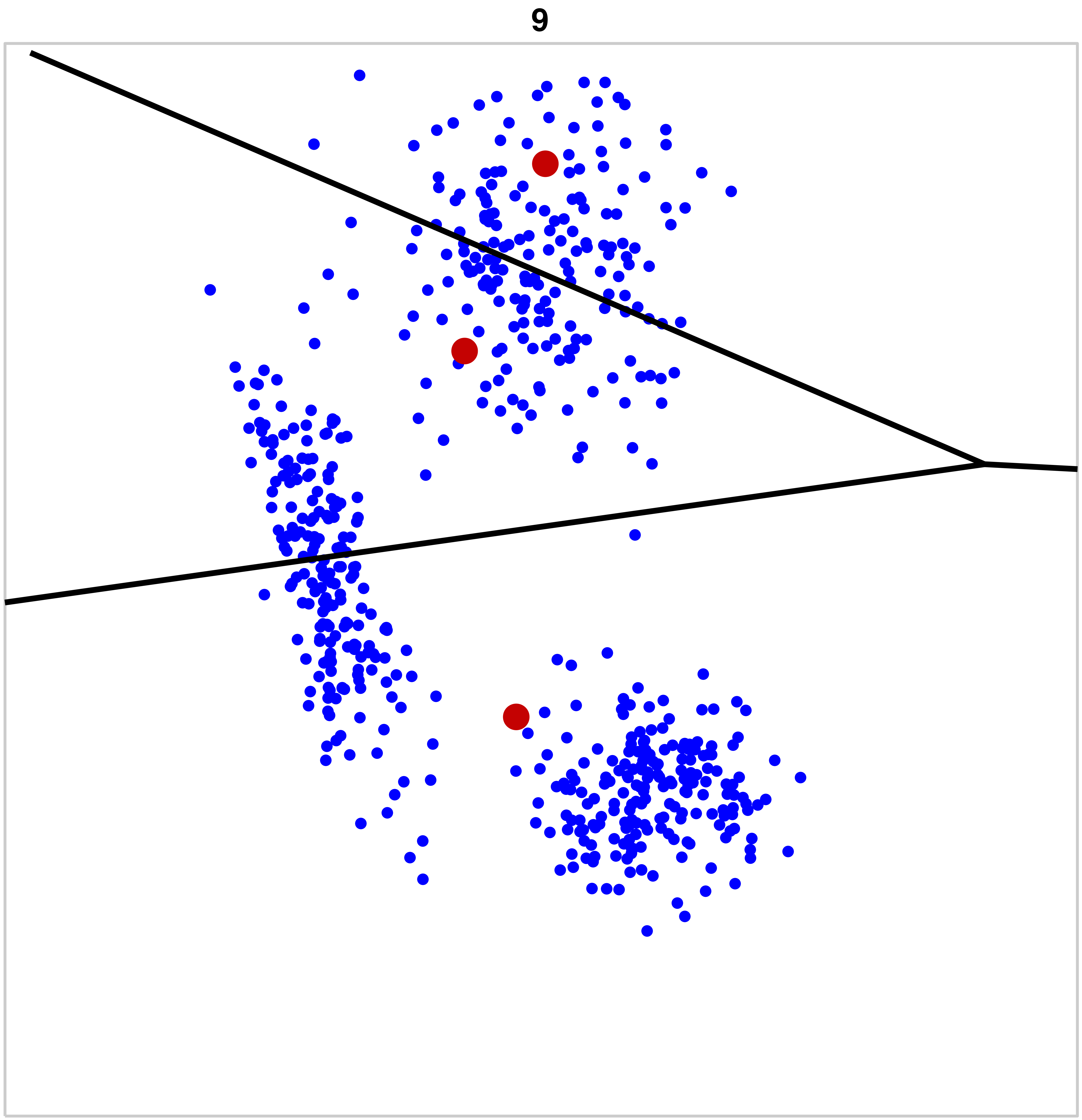
Find center of each cluster



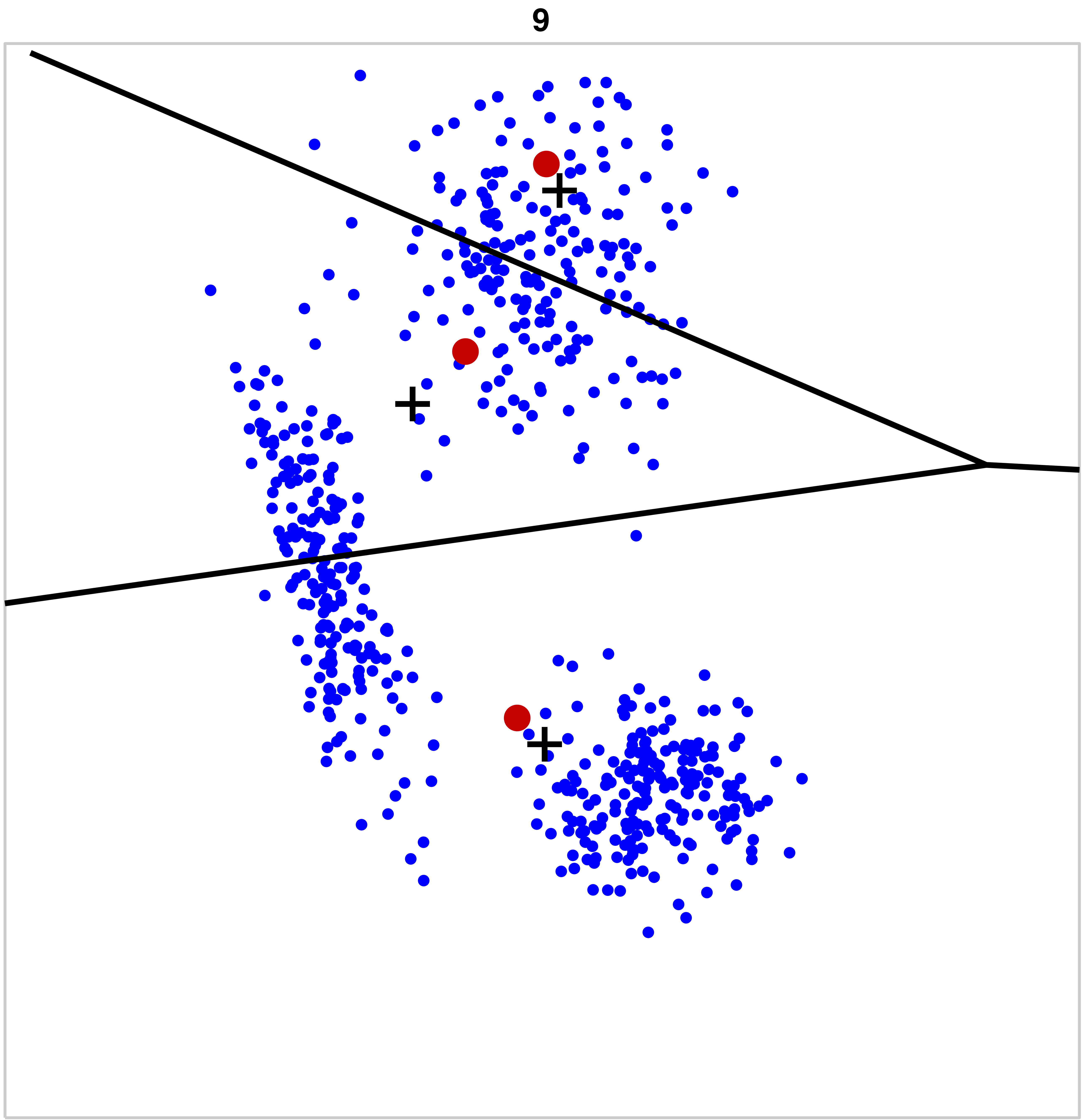
Update the mean



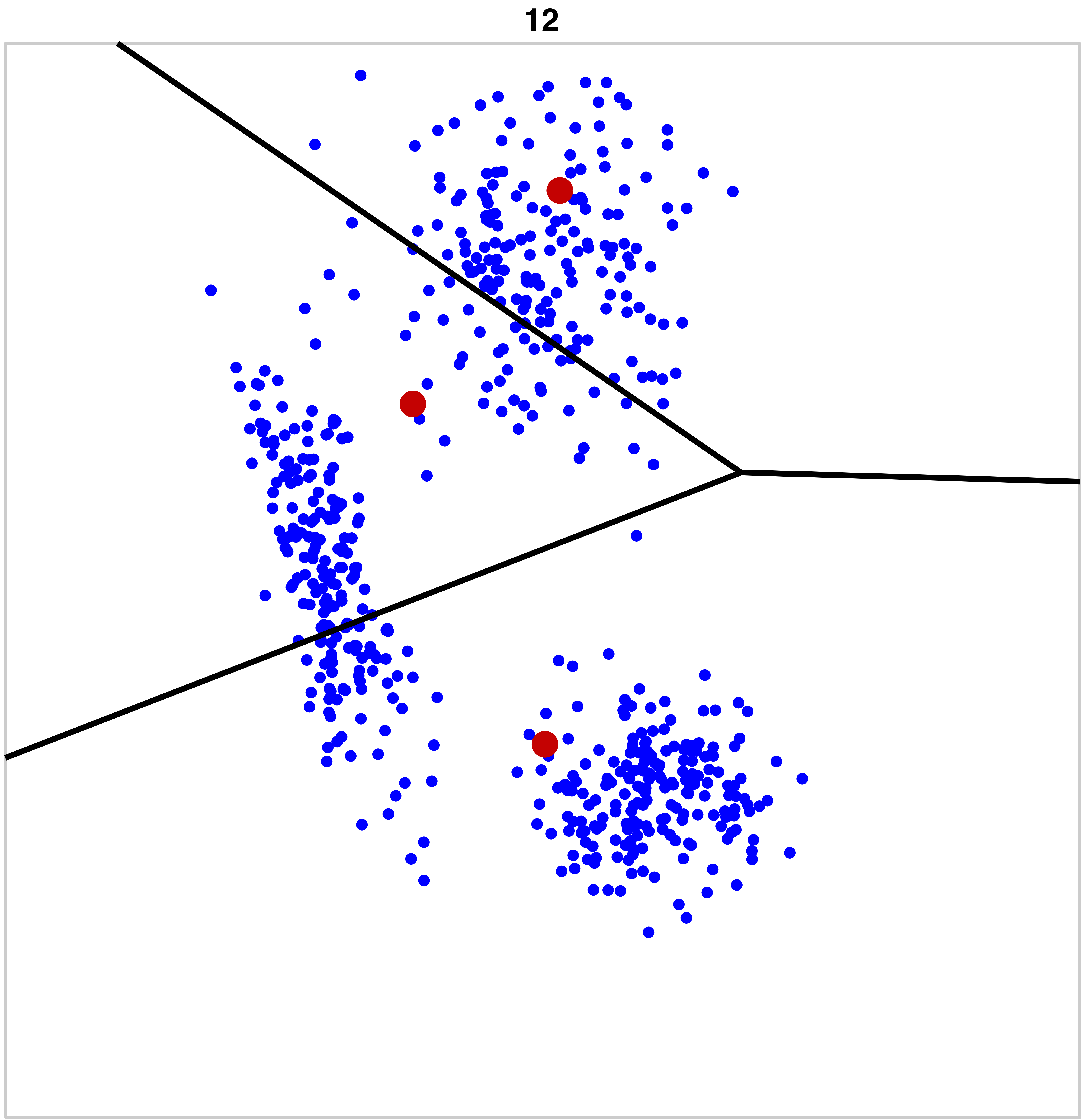
Find reconstruction regions



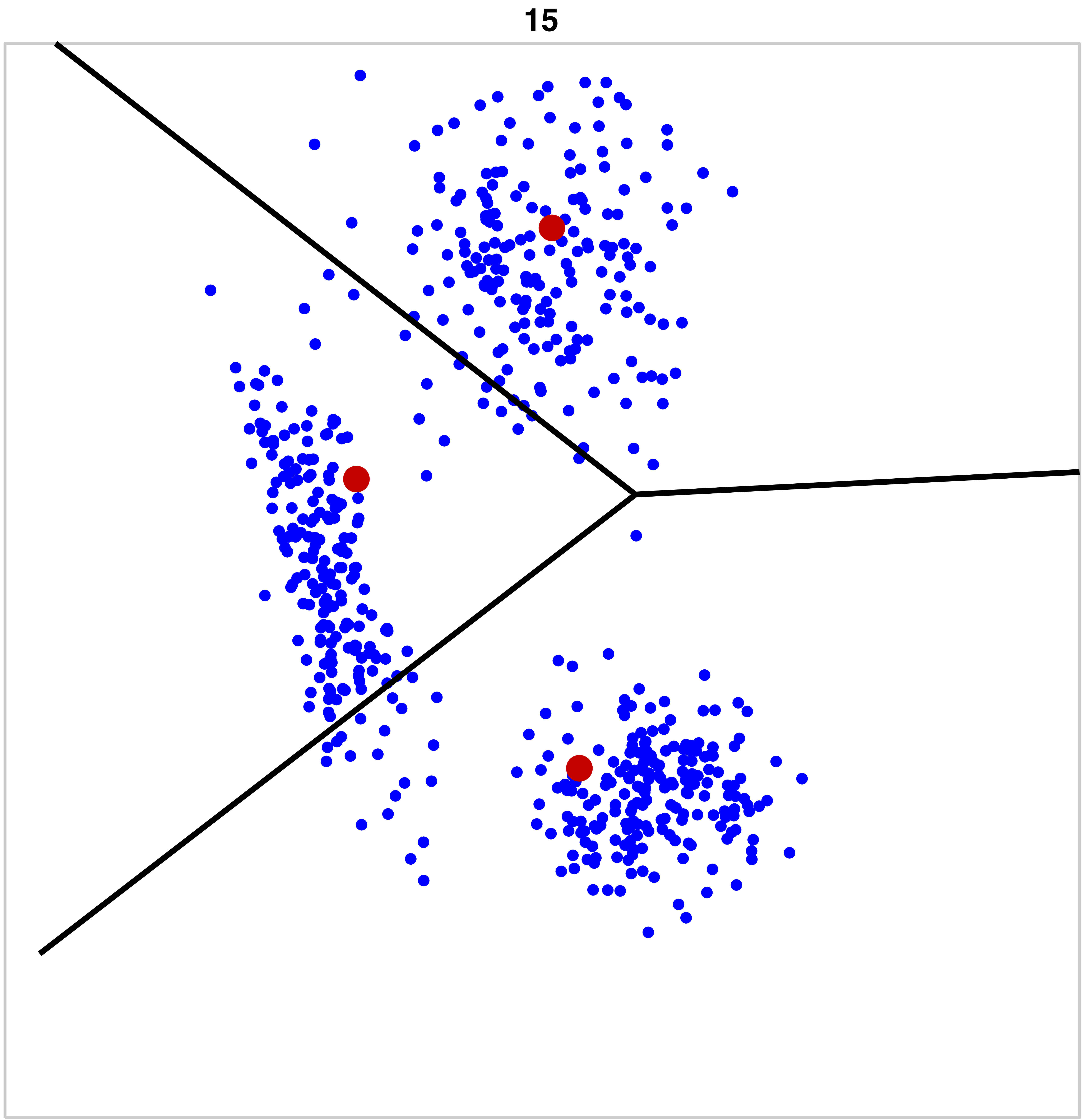
Find center of each cluster



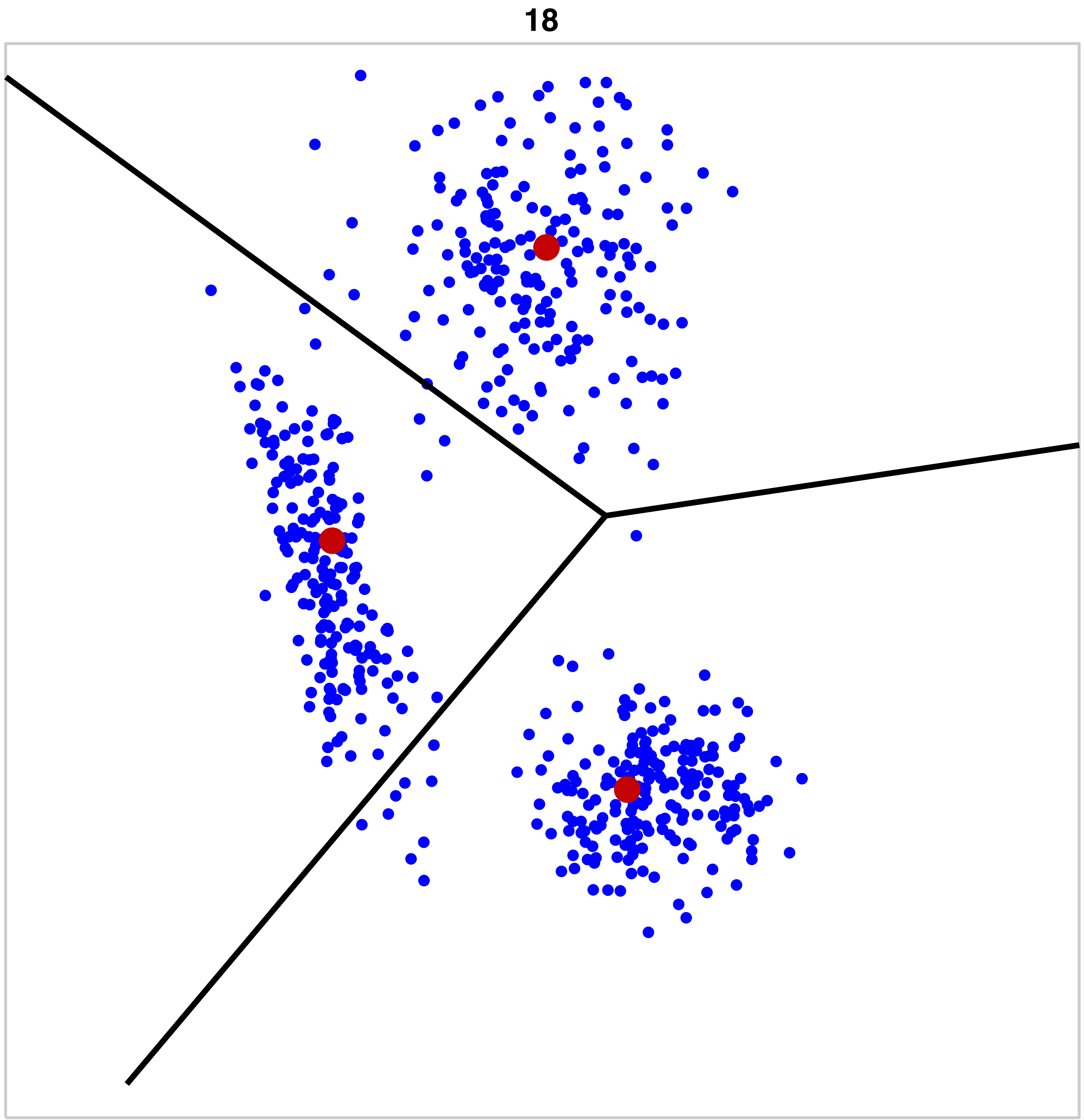
Update the mean, and update the reconstruction region. Now you get the idea.



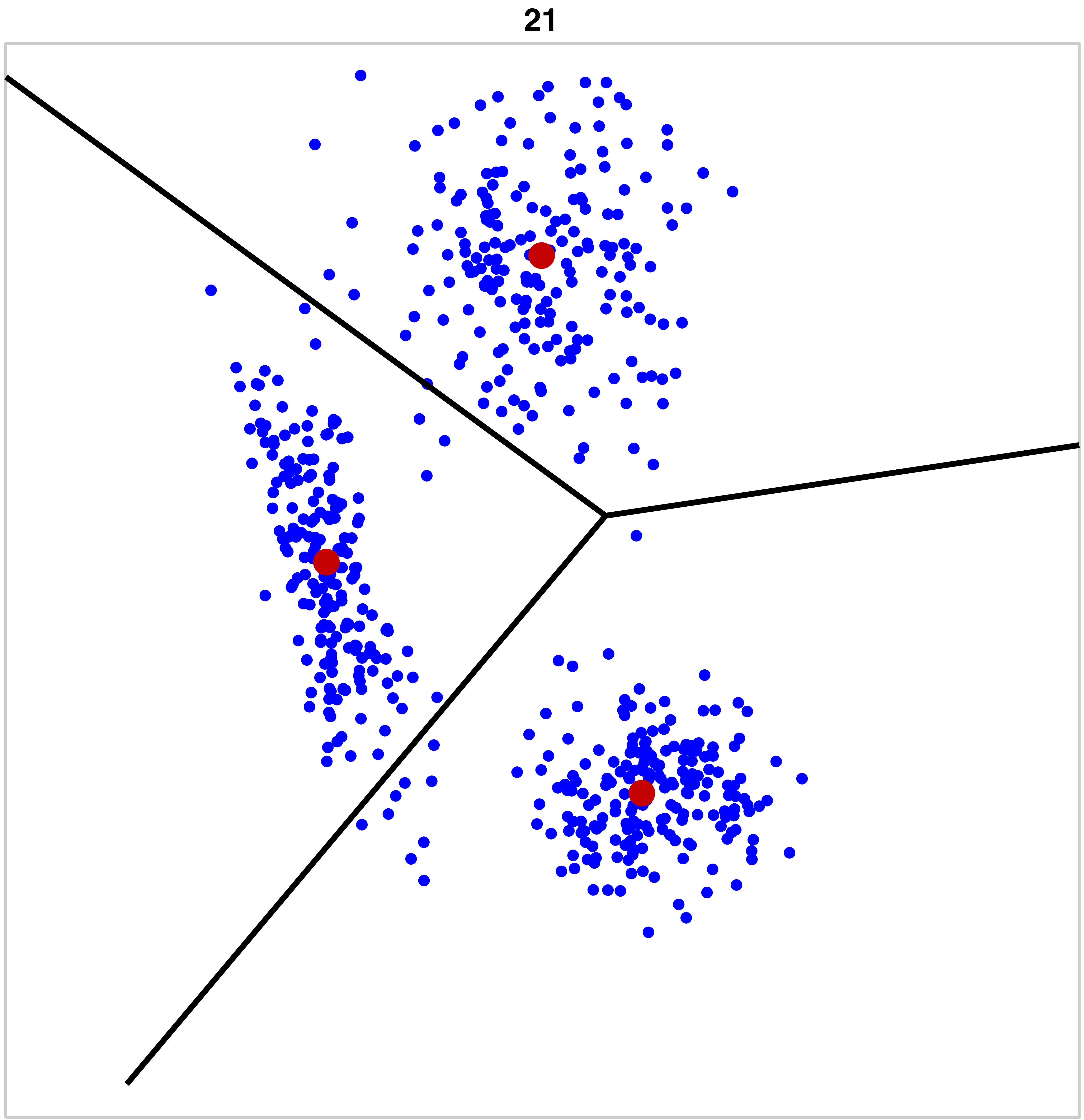
Iteration 5



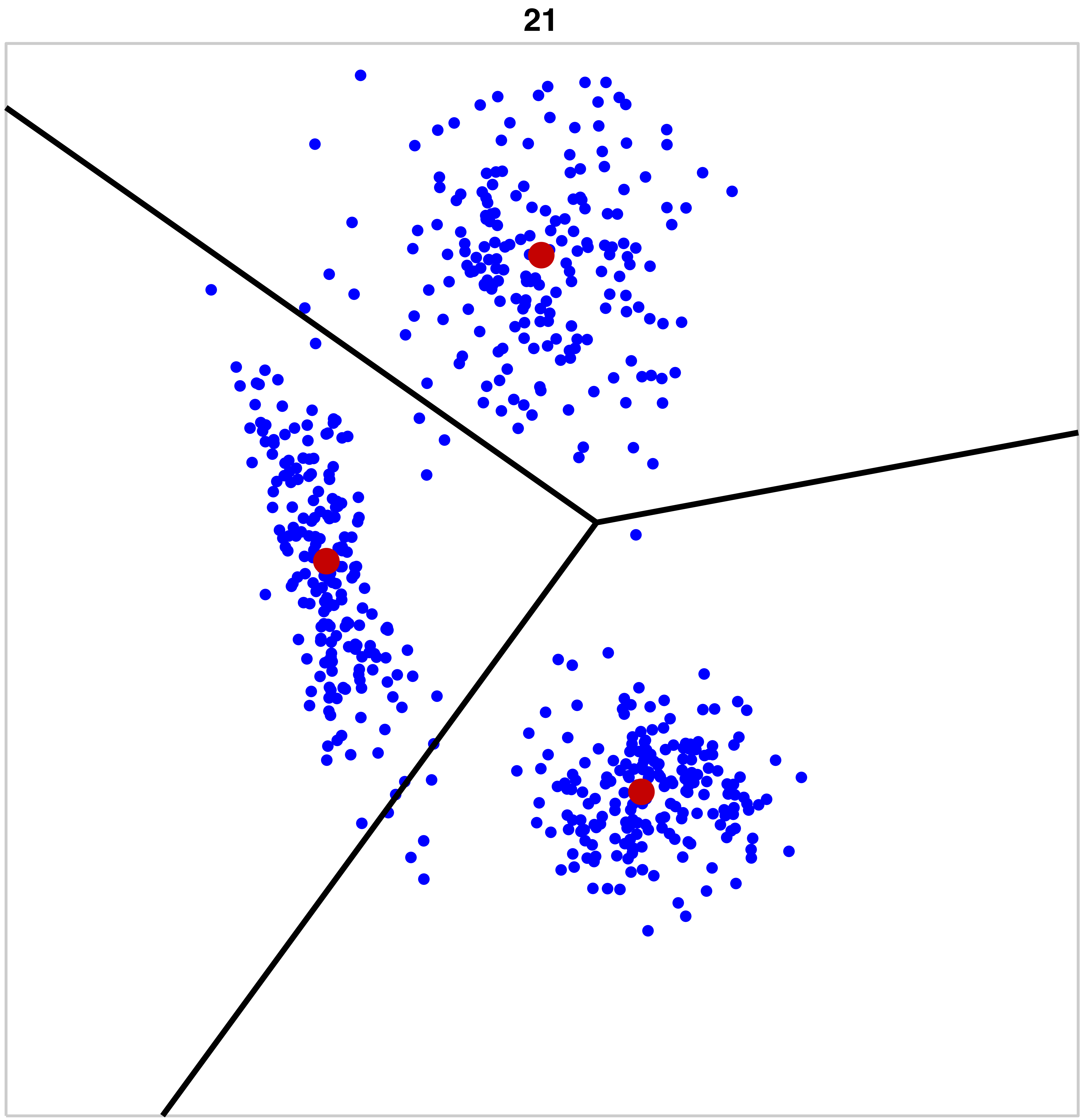
Iteration 6



Iteration 7



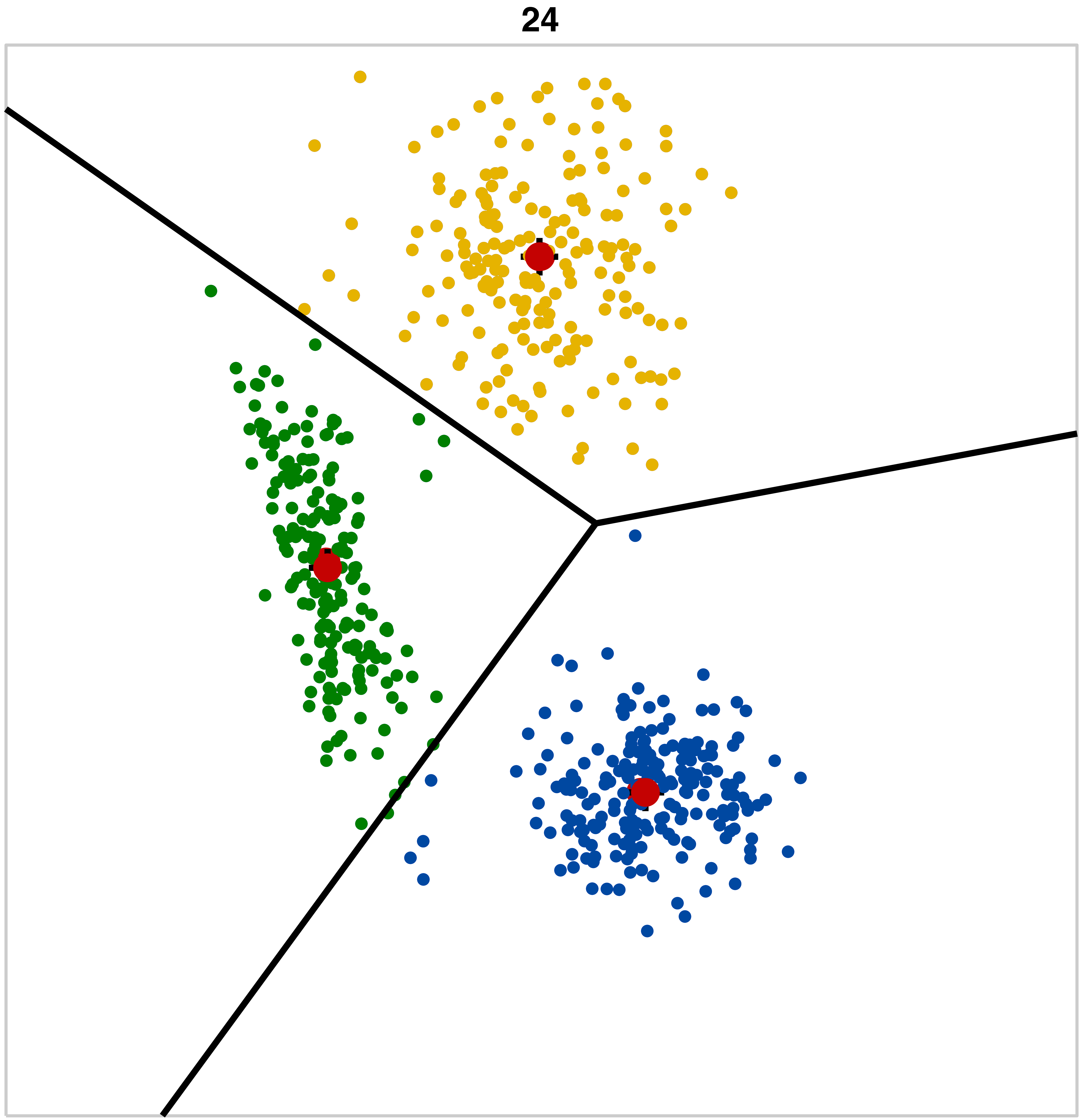
Iteration 8



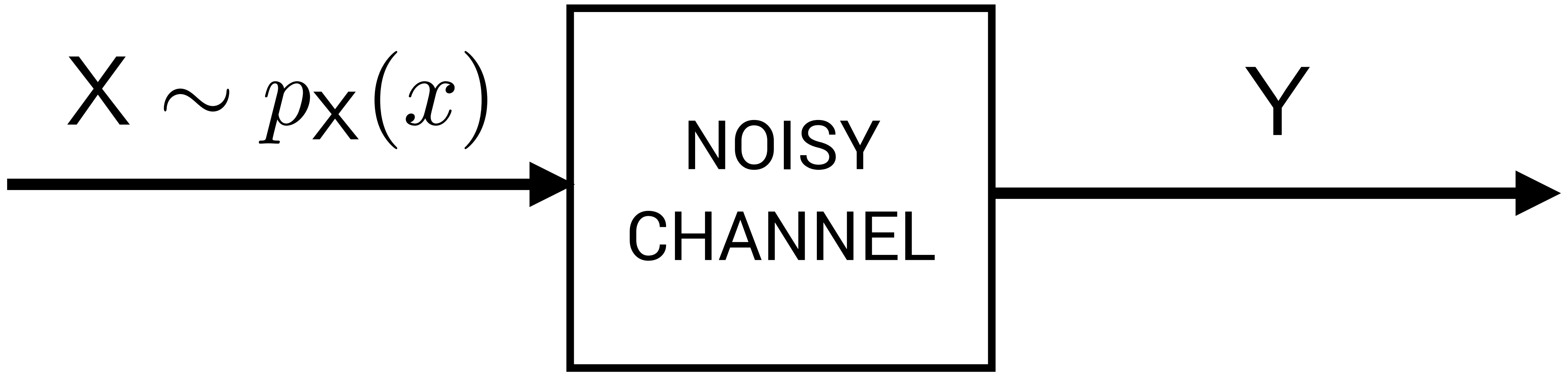
Finished:

Each point is classified into one of three regions.

Good approximation of the underlying classes.



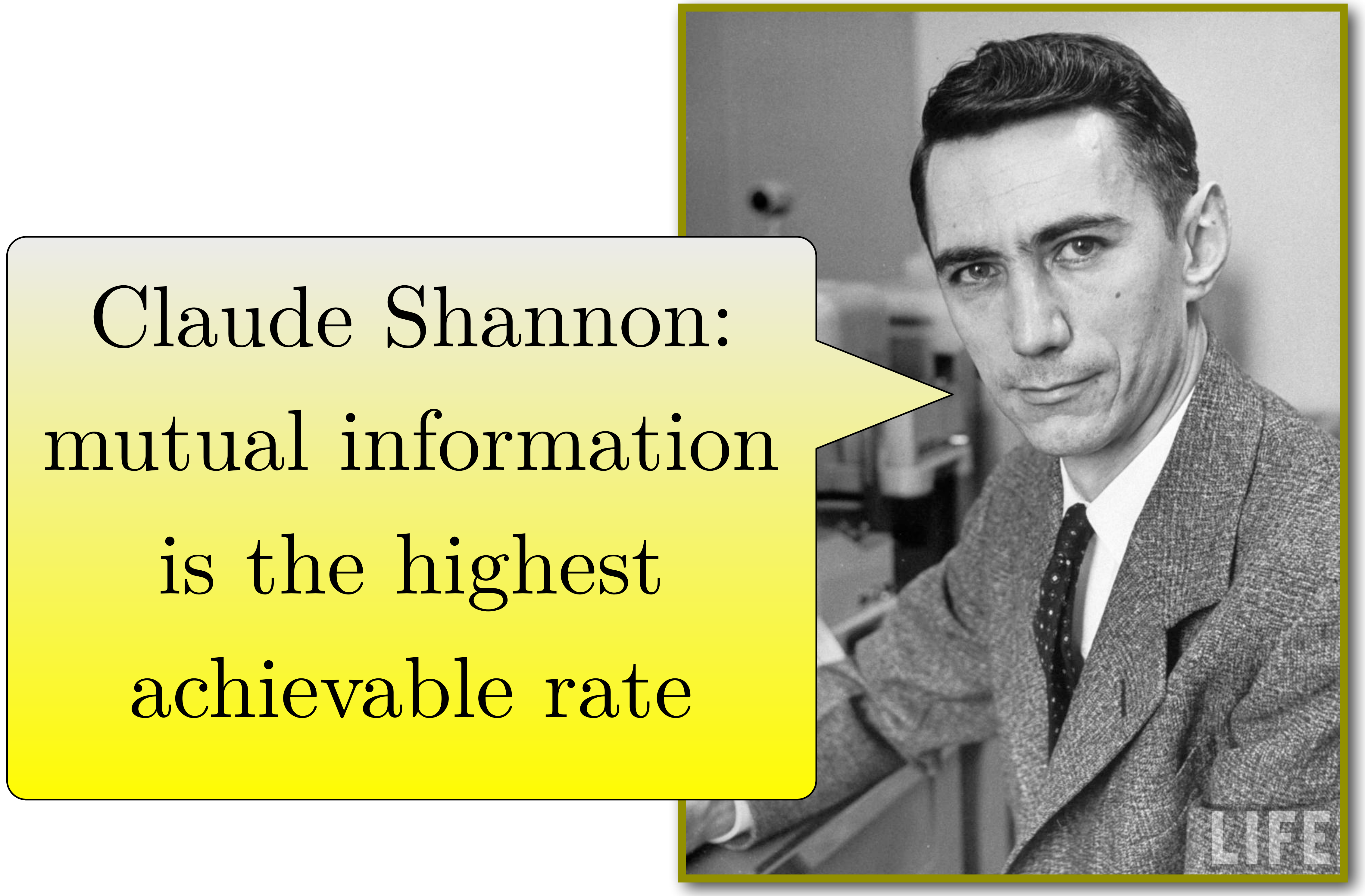
Just Enough Information Theory



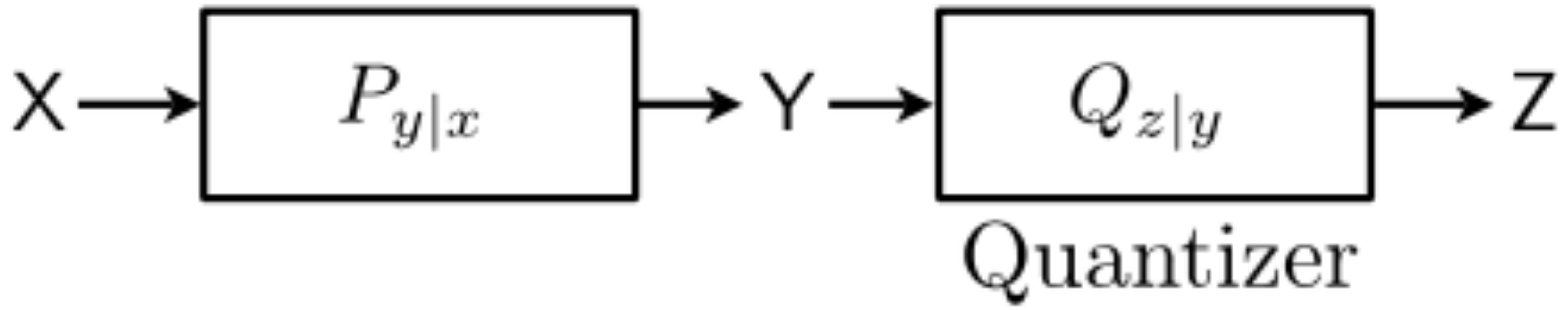
What is the best code we can design?

$$R < C = \max_{p_X(x)} I(X;Y)$$

Code rate < Channel Capacity



Highest Achievable Rate for Communications over a Quantized Channel



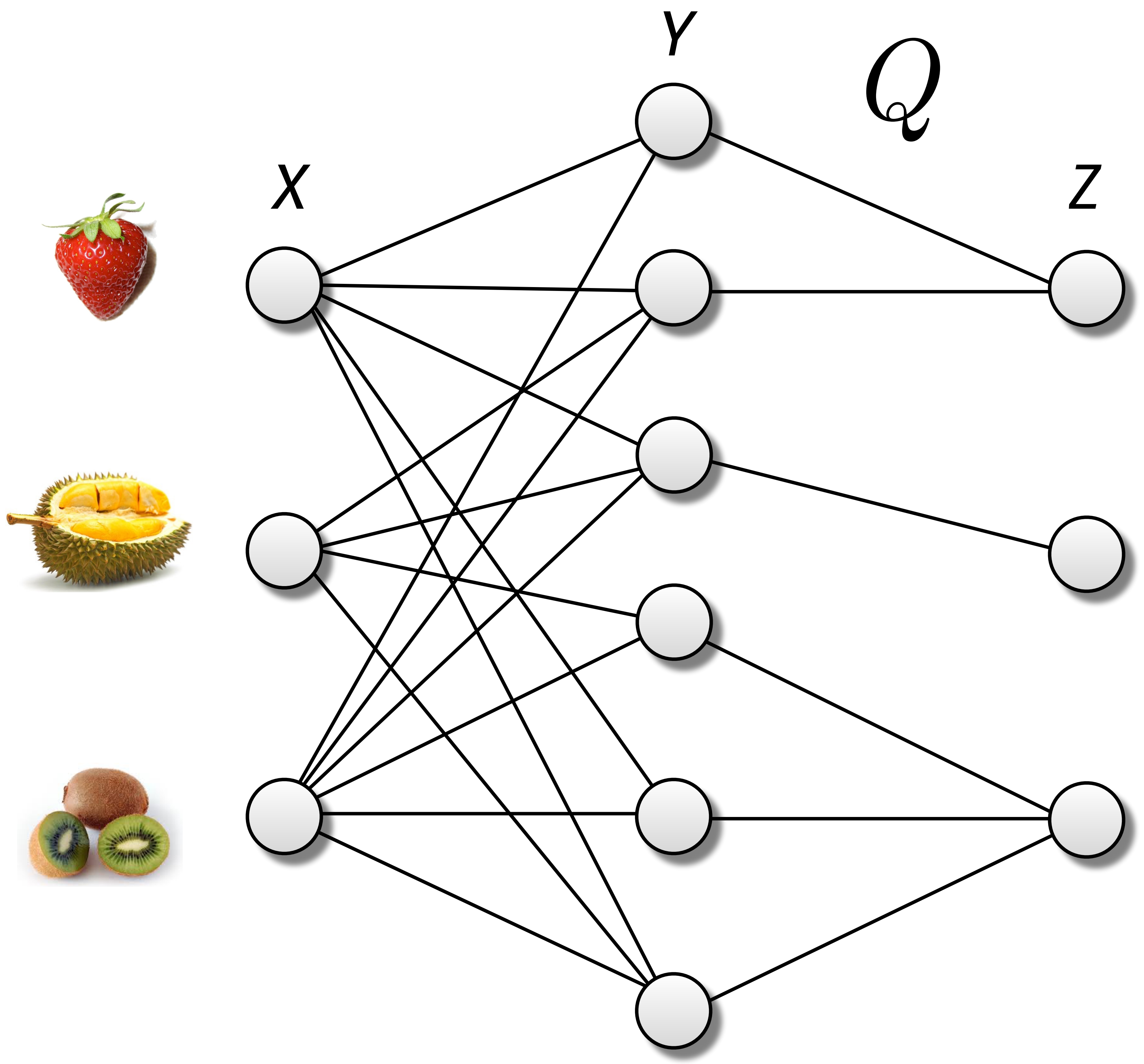
Given a channel, find the *quantizer* Q which maximizes the achievable rate:

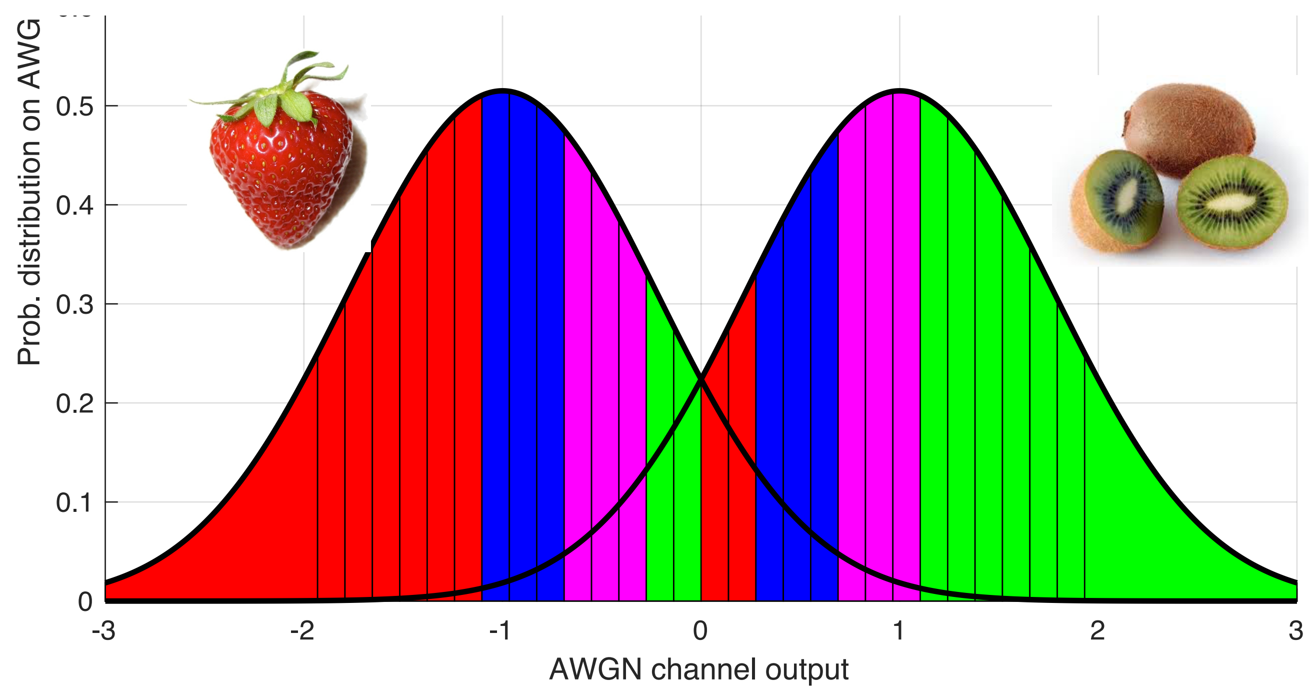
$$C = \max_{Q} I(\mathsf{X};\mathsf{Z})$$

We will fix the input distribution $p_{\mathsf{X}}(x)$.

Jointly optimizing Q and $p_{\mathsf{X}}(x)$ is a much more difficult problem.

Connecting Classification and Quantization





$$\mathsf{X} \xrightarrow{\Pr(\mathsf{Y}|\mathsf{X})} \mathsf{Y} \xrightarrow{Q} \mathsf{Z}$$

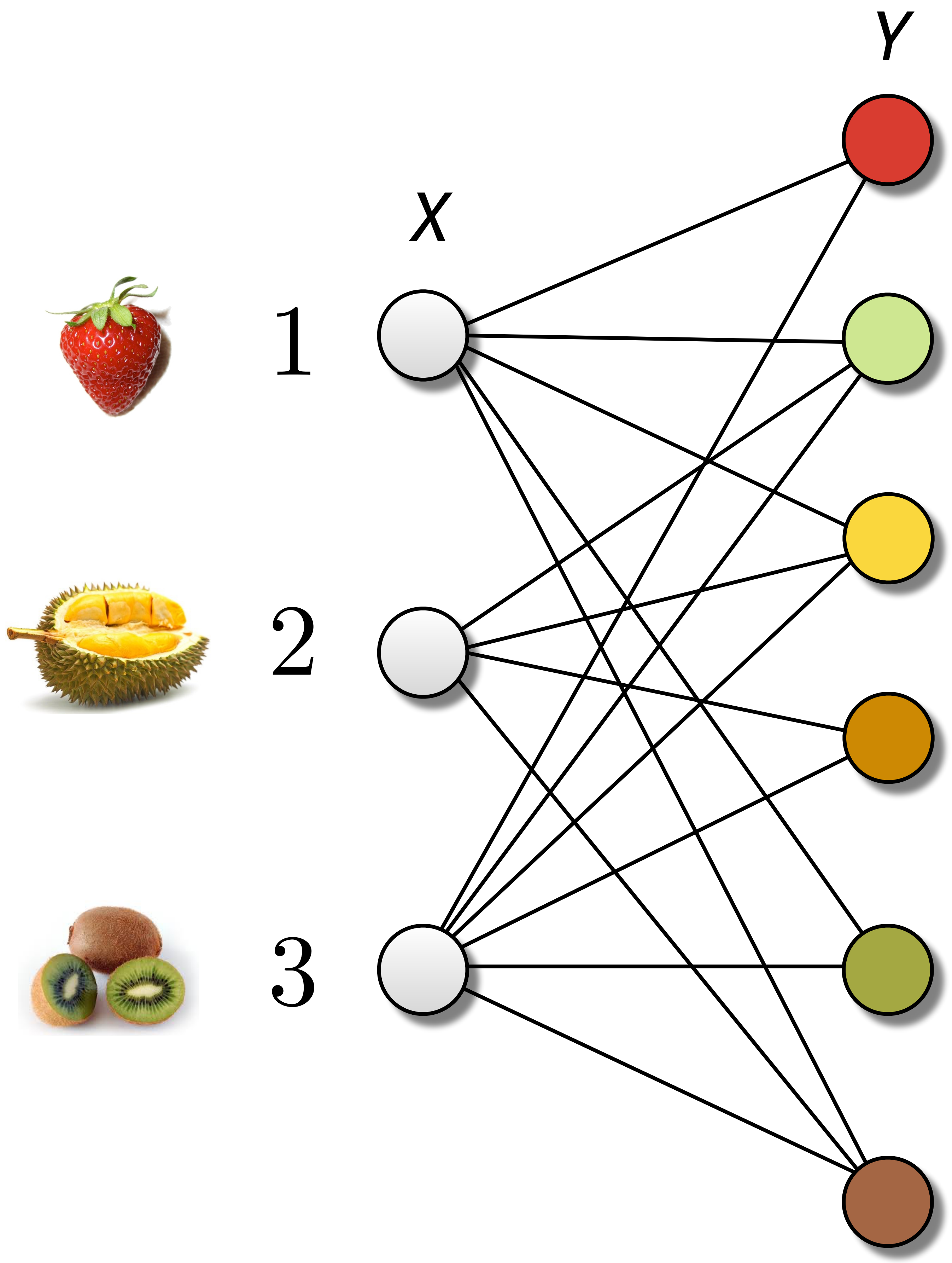
Given a discrete memoryless channel and input distribution $p_{\mathsf{XY}}(x, y)$, find the quantizer Q which maximizes mutual information:

$$Q^* = \arg\max_Q I(\mathsf{X}; \mathsf{Z})$$

with $|\mathsf{Z}| < |\mathsf{Y}|$.

$|\mathsf{Z}| \geq |\mathsf{Y}|$ is trivial.

Problem Setup and Backwards Channel



Assume $p_{\mathsf{XY}}(x, y)$ is known; X is discrete. Examples show Y is discrete, but results can be extend to continuous case. Running example:

$$\mathsf{X} \in \{1, 2, 3\}$$
$$\mathsf{Y} \in \{\text{red}, \text{lime}, \text{yellow}, \text{orange}, \text{green}, \text{brown}\}$$

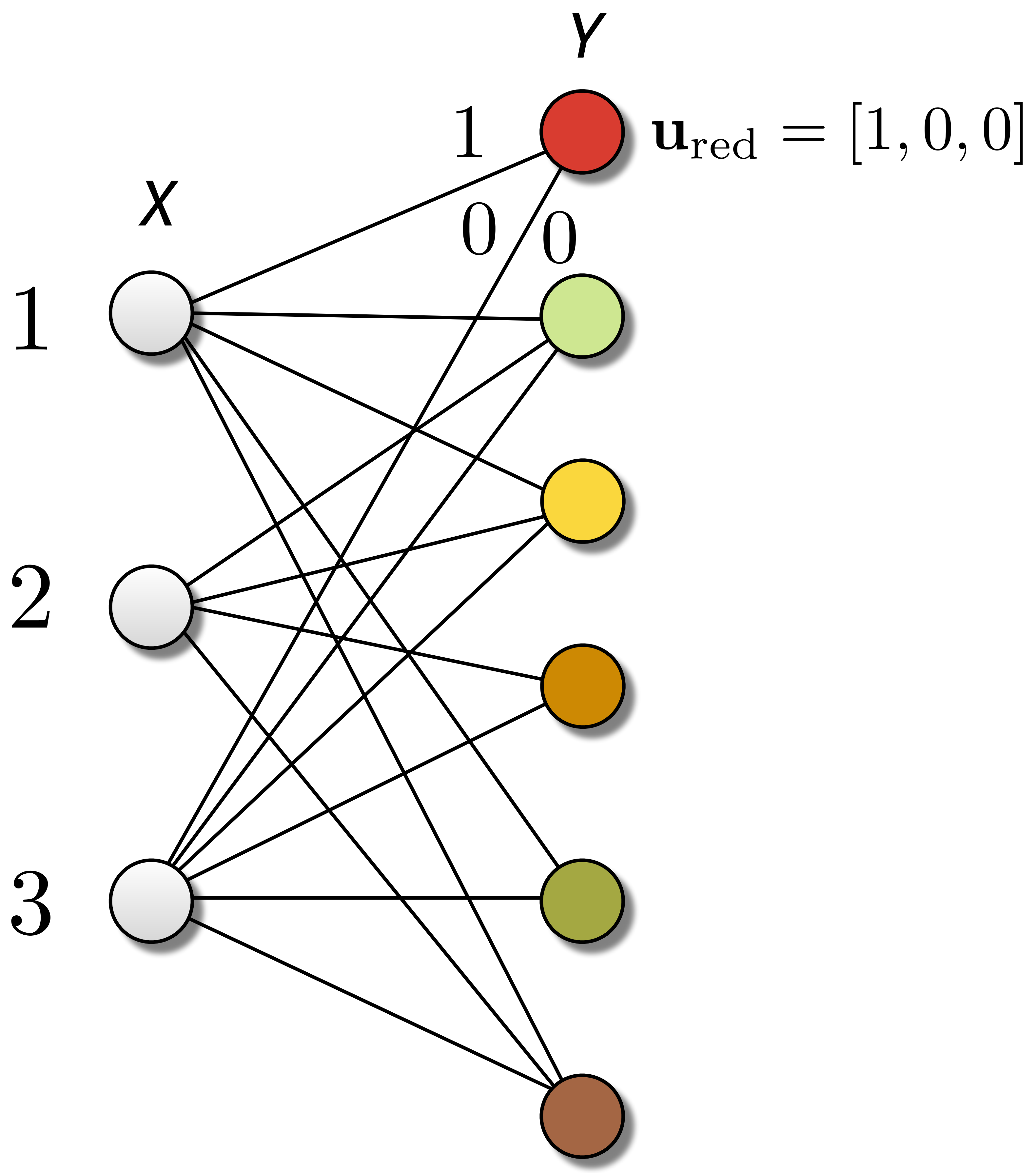
Work with the backward channel:

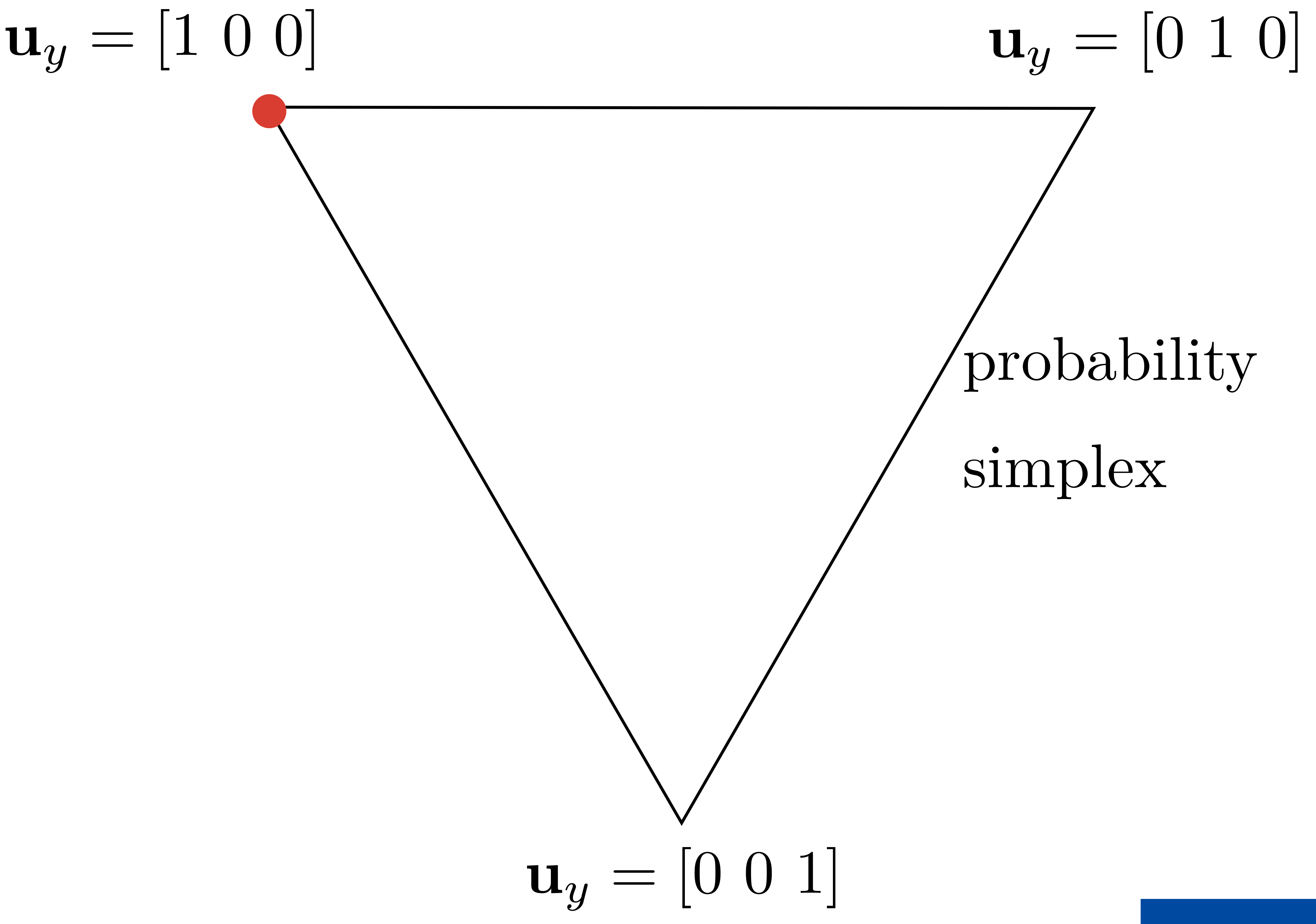
$$\mathbf{u}_y = \left[\Pr(\mathsf{X} = 1|\mathsf{Y} = y), \ldots, \Pr(\mathsf{X} = J|\mathsf{Y} = y)\right]$$

Justify this later.

Backward Channel $\Pr(\mathsf{X} \mid \mathsf{Y})$ as a Vector

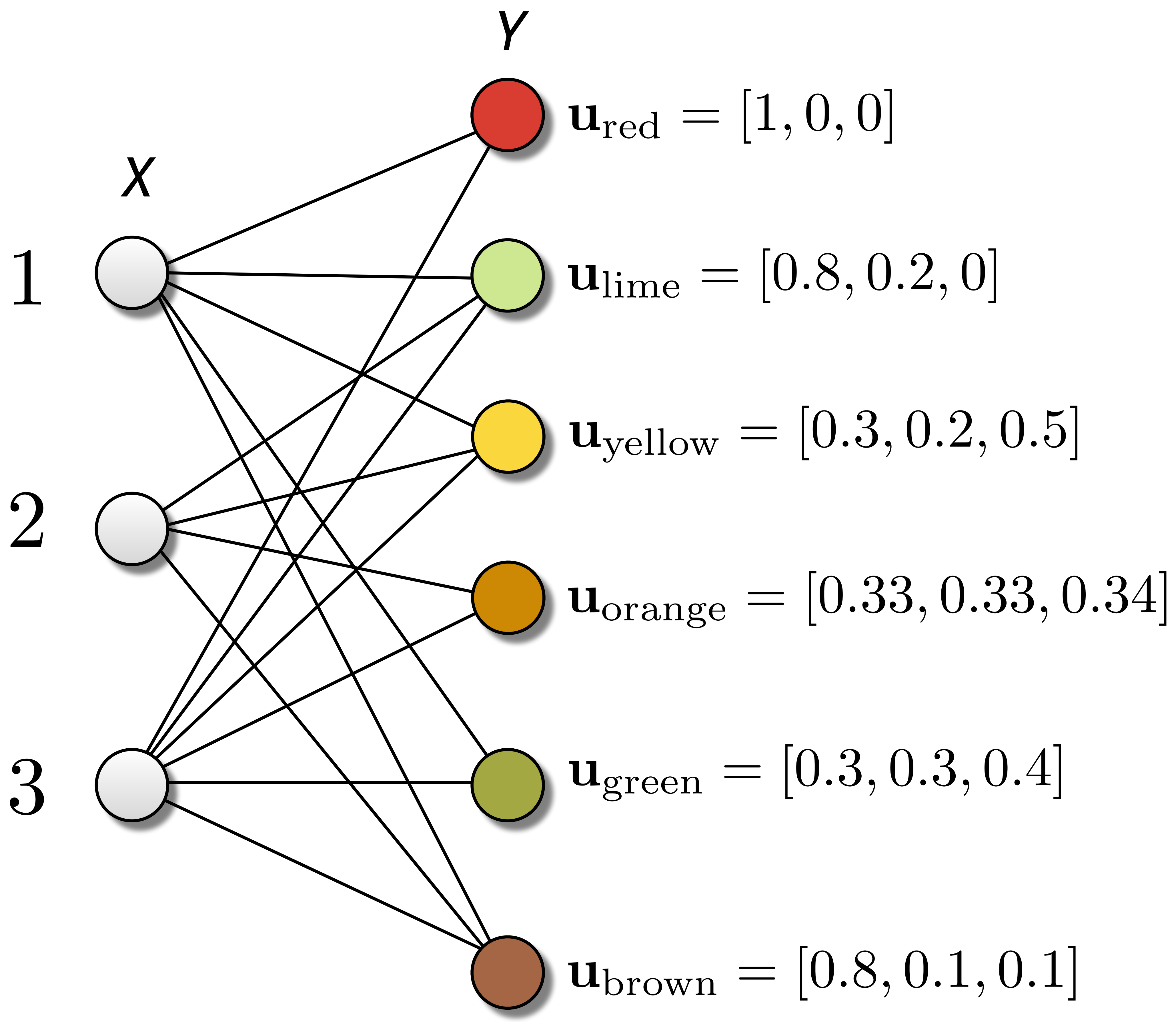
$$\mathbf{u}_y = \Big[\Pr(\mathsf{X} = 1 | \mathsf{Y} = y), \Pr(\mathsf{X} = 2 | \mathsf{Y} = y), \Pr(\mathsf{X} = 3 | \mathsf{Y} = y) \Big]$$

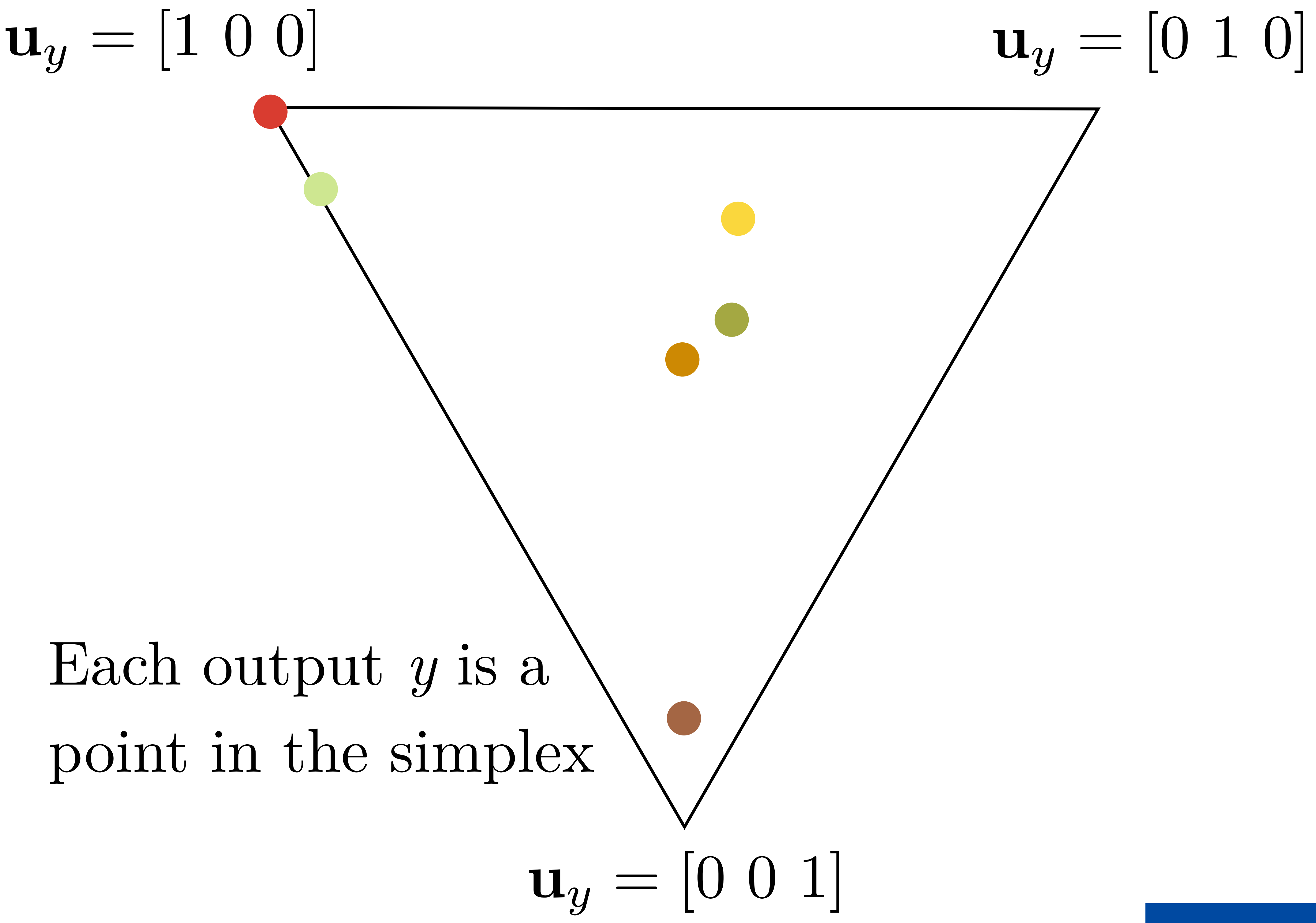




Backward Channel $\Pr(X \mid Y)$ as a Vector

$$\mathbf{u}_y = \Big[\Pr(X = 1|Y = y), \Pr(X = 2|Y = y), \Pr(X = 3|Y = y)\Big]$$

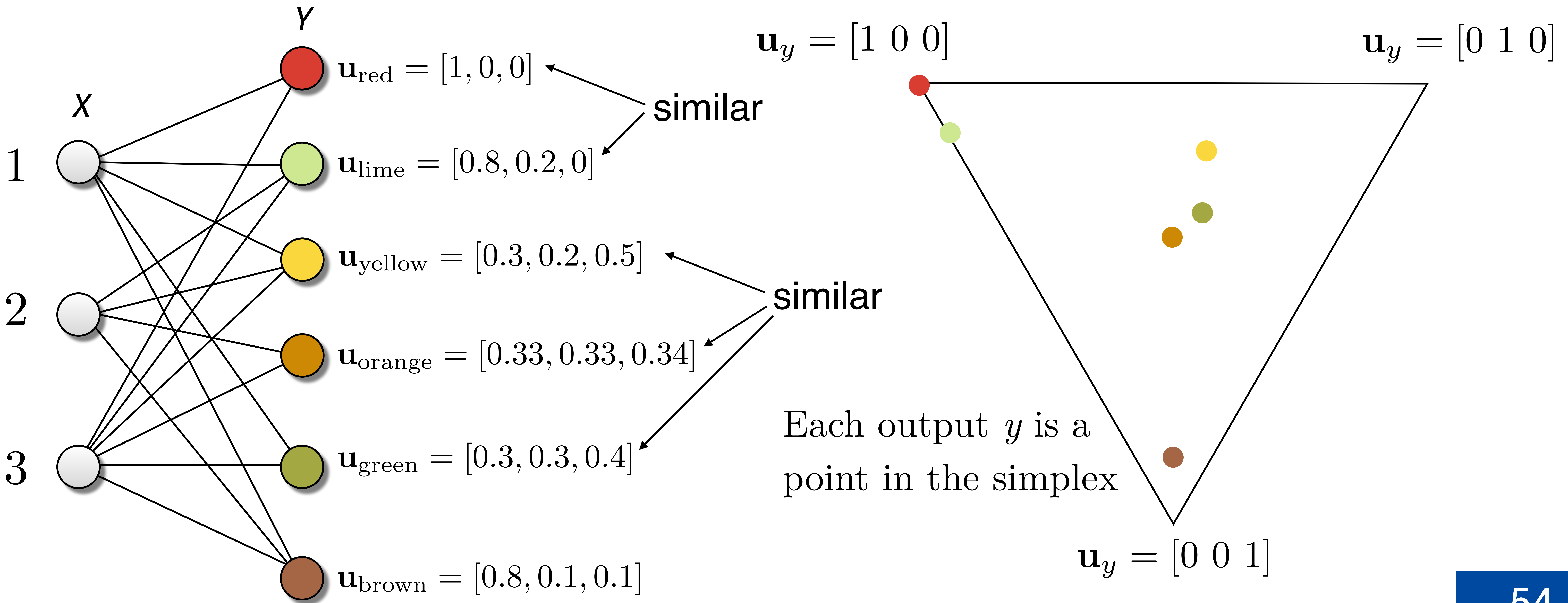




Each output y is a point in the simplex

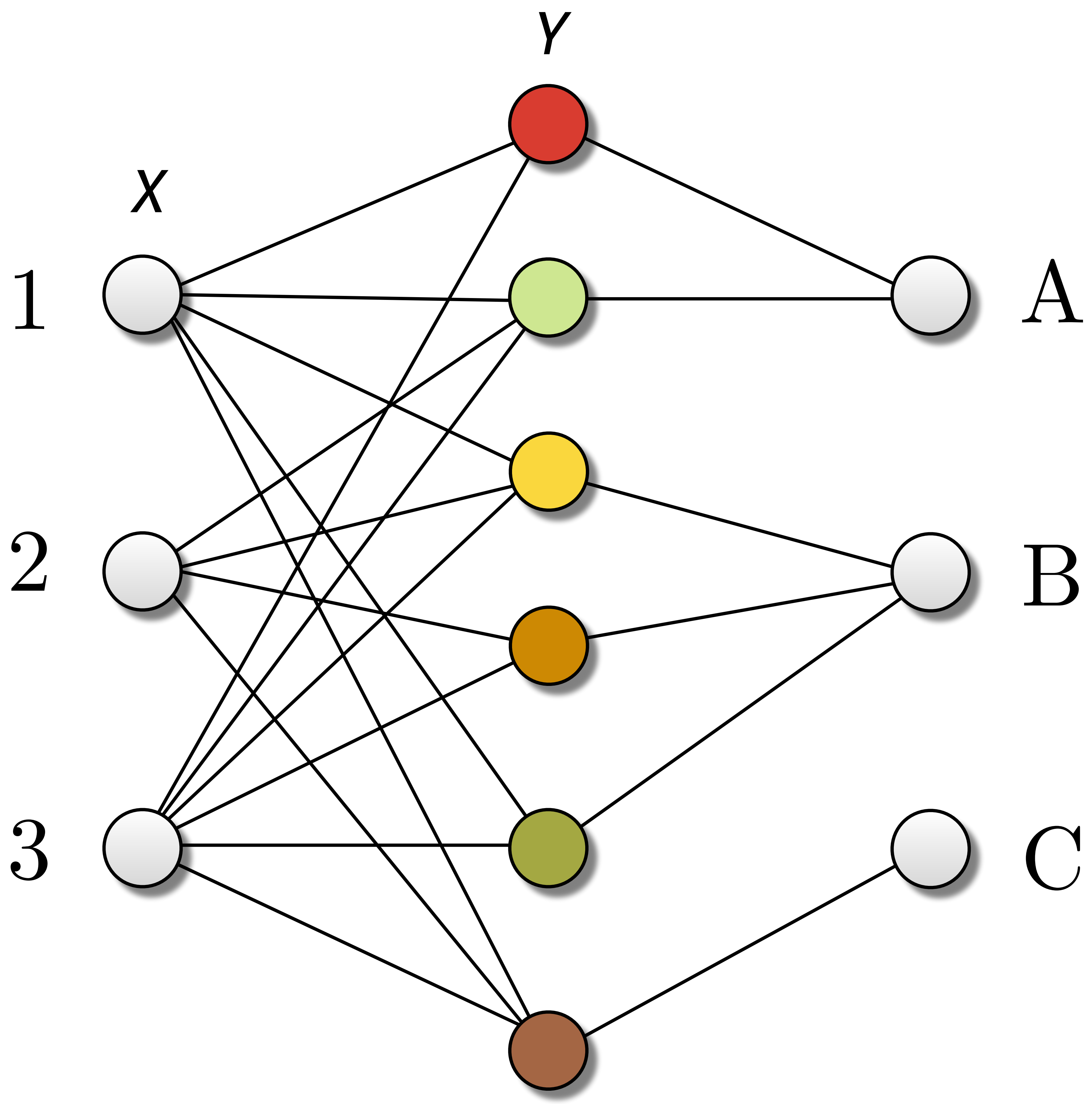
Backward Channel $\Pr(X \mid Y)$ as a Vector

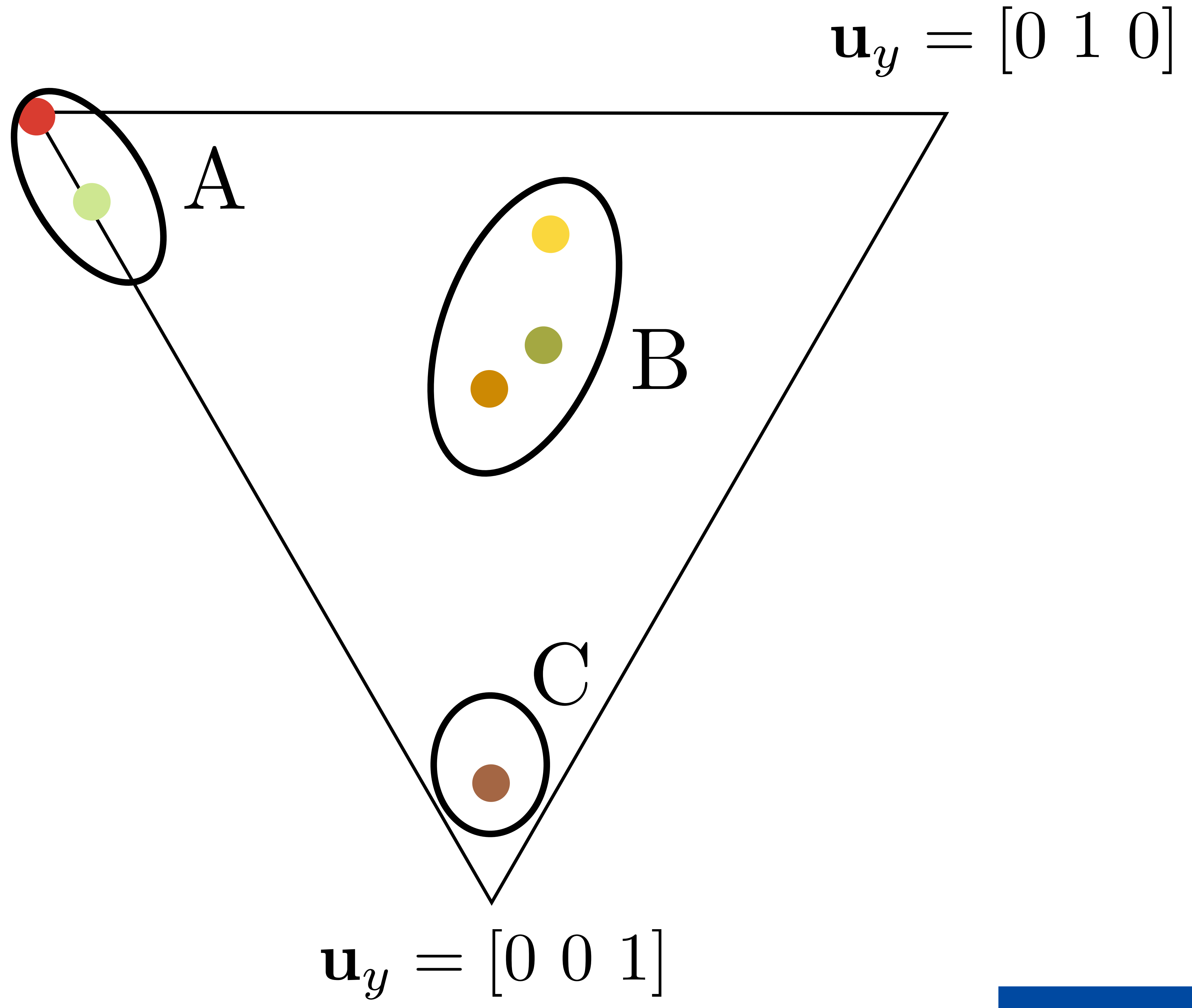
$$\mathbf{u}_y = \left[\Pr(X = 1|Y = y), \Pr(X = 2|Y = y), \Pr(X = 3|Y = y)\right]$$



Backward Channel $\Pr(X \mid Y)$ as a Vector

$$\mathbf{u}_y = \left[\Pr(X = 1|Y = y), \Pr(X = 2|Y = y), \Pr(X = 3|Y = y) \right]$$

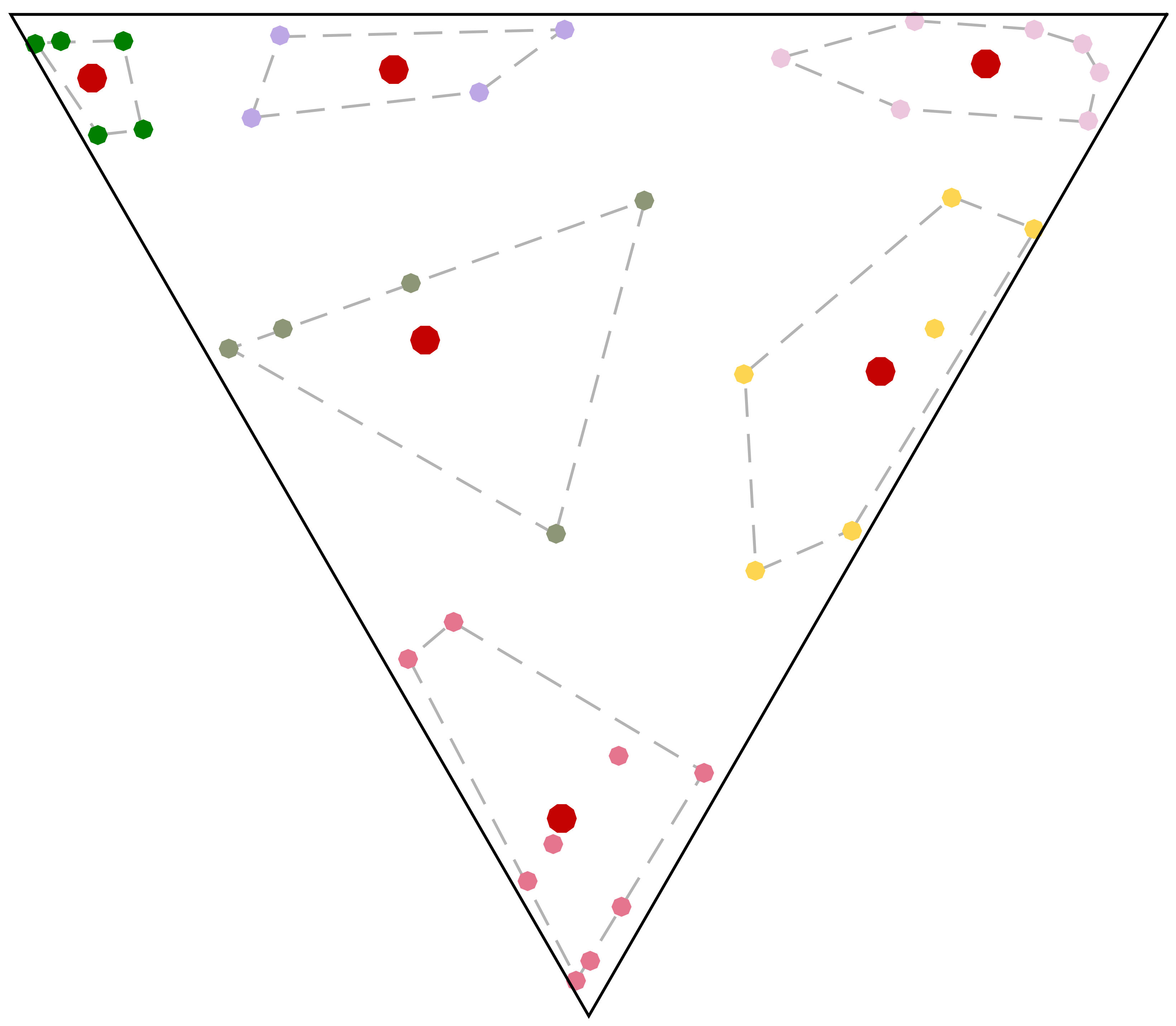




K-Means With KL Divergence Metric

"KL-Means algorithm" replace Euclidean distance with KL distance

$$\mathbf{U} = \left[\Pr(\mathsf{X} = 1|\mathsf{Y}), \ldots, \Pr(\mathsf{X} = J|\mathsf{Y})\right]$$
$$\mathbf{V} = \left[\Pr(\mathsf{X} = 1|\mathsf{Z}), \ldots, \Pr(\mathsf{X} = J|\mathsf{Z})\right]$$



Then, the following holds:

$$I(\mathsf{X};\mathsf{Y}) - I(\mathsf{X};\mathsf{Z}) = E\big(D(\mathbf{U}||\mathbf{V})\big)$$

$D(\cdot||\cdot)$ is the Kullback-Leiber divergence

$$Q^* = \arg\max_{Q} I(\mathsf{X};\mathsf{Z}) = \arg\min_{Q} E\big(D(\mathbf{U}||\mathbf{V})\big)$$

Thus, maximization of mutual information is minimization of KL divergence.

K-Means With KL Divergence Metric

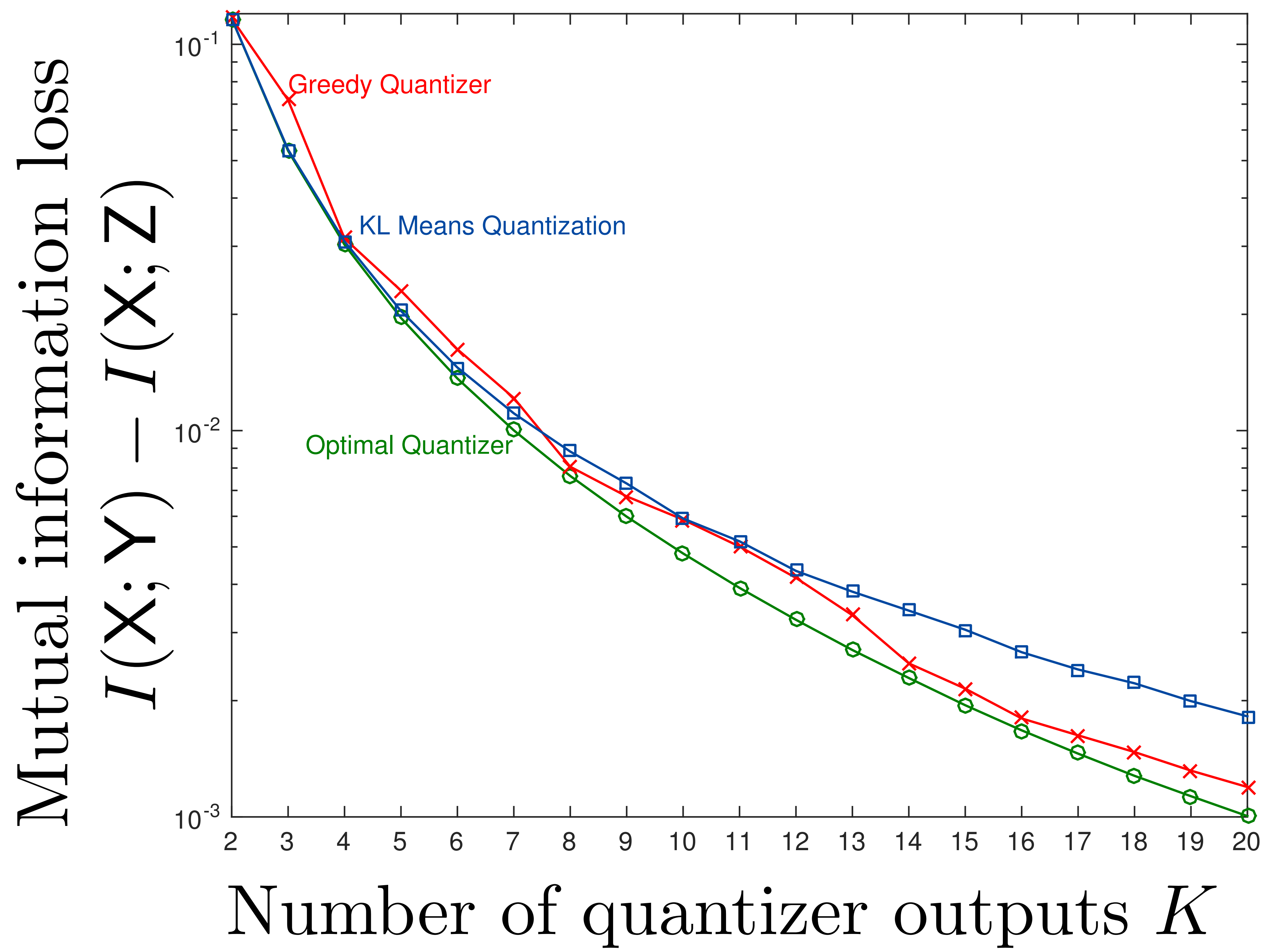
"KL-Means algorithm" replace Euclidean distance with KL distance

Min KL divergence = max. mutual information

$$Q^* = \arg\max_Q I(\mathsf{X};\mathsf{Z}) = \arg\min_Q E\Big(D(\mathbf{U}||\mathbf{V})\Big)$$

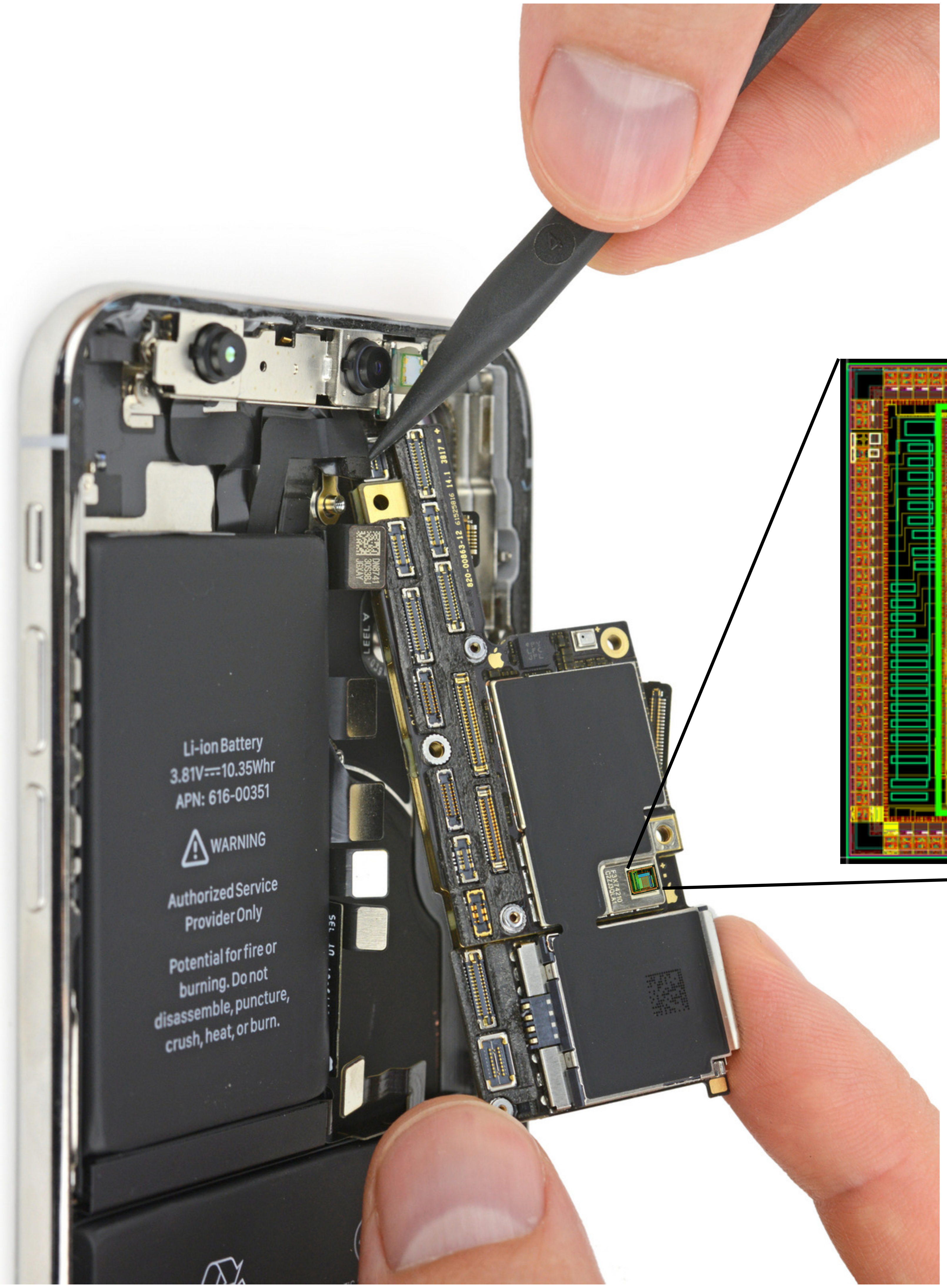
Numerical results show tradeoff:

- increasing number of quantizer outputs
- decreases the loss of mutual information

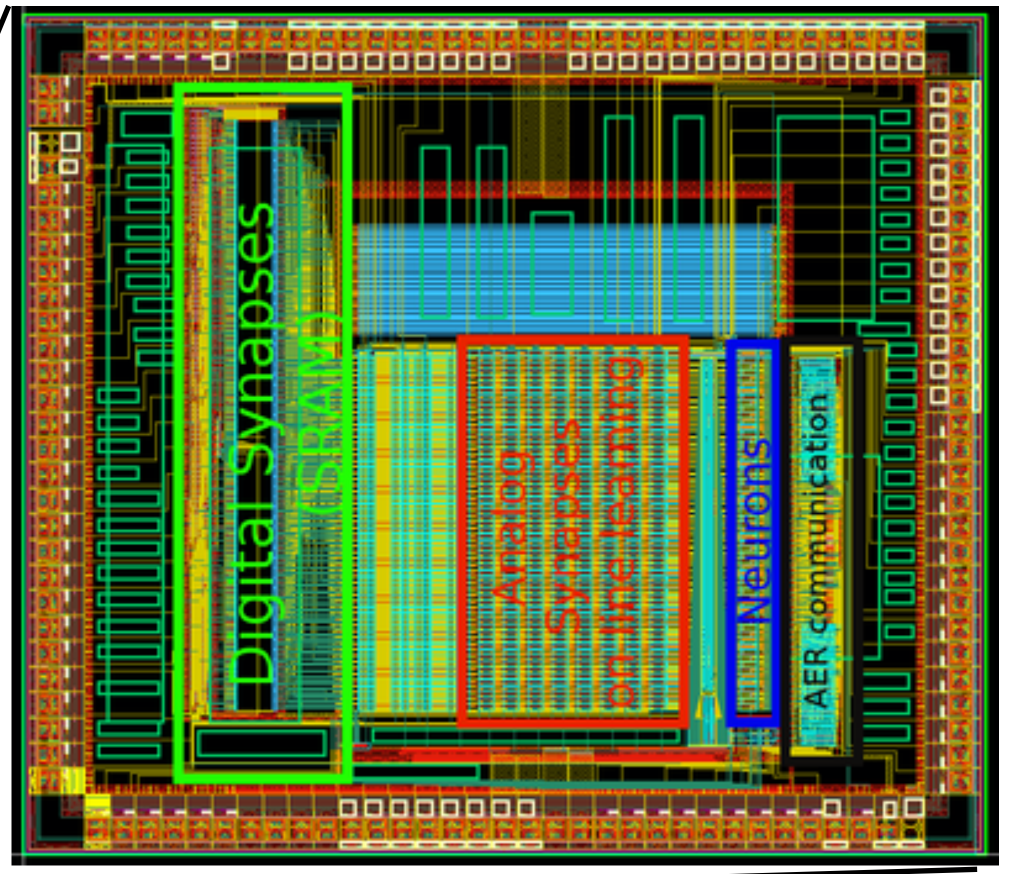


A. Zhang and B. Kurkoski, "Low-Complexity Quantization of Non-Binary Input DMCs" ISITA 2016.

Motivation 3: LDPC Decoding for VLSI



VLSI



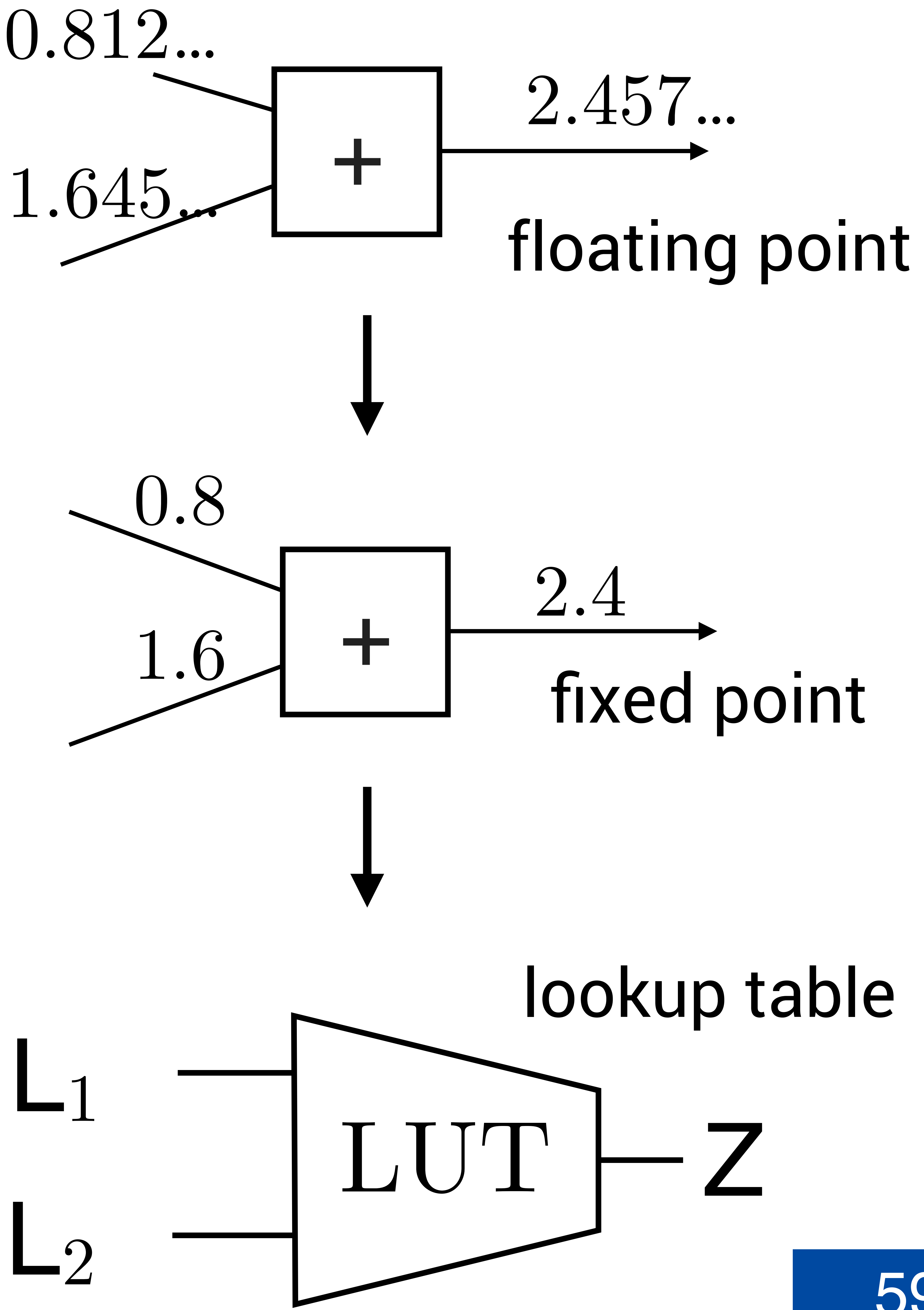
- LDPC deoders implemented in VLSI
- VLSI approximates floating point numbers using fixed point
- tradeoff between number of bits and performance
- “ad hoc” implementation by engineers

ifixit.com

ETH Zurich http://bit.ly/2nTEfCy

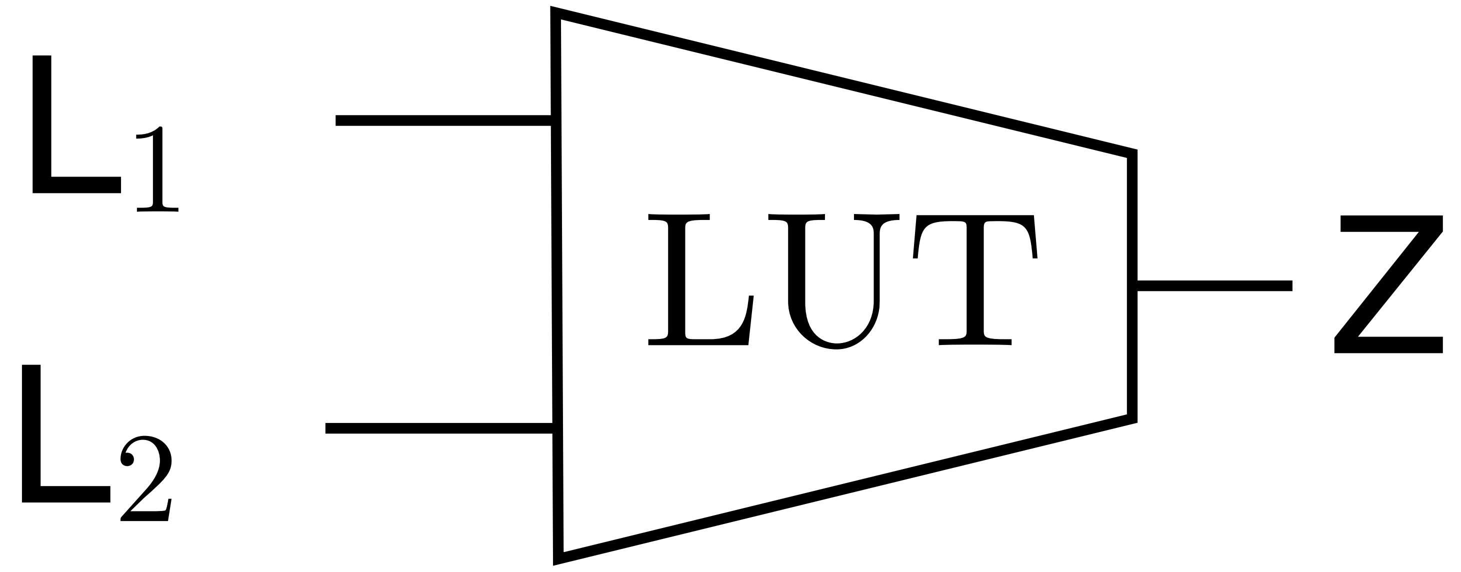
Max-LUT Method

Max-LUT is a method for implementing fixed-point LDPC decoders, using lookup tables that maximize mutual information.



Max-LUT Method: Central Idea

Decoder Node



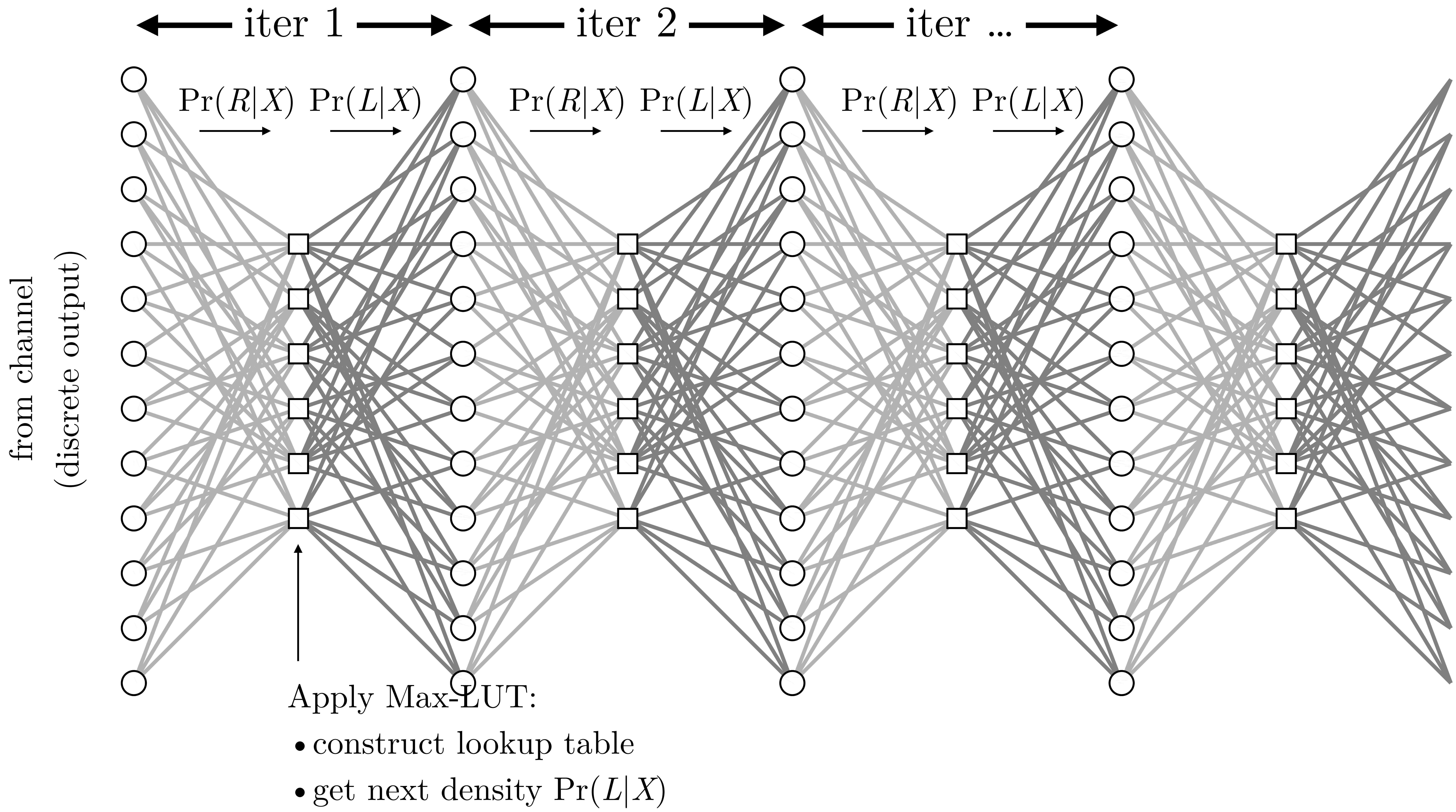
L_i are noisy versions of X

Z is a noisy version of X

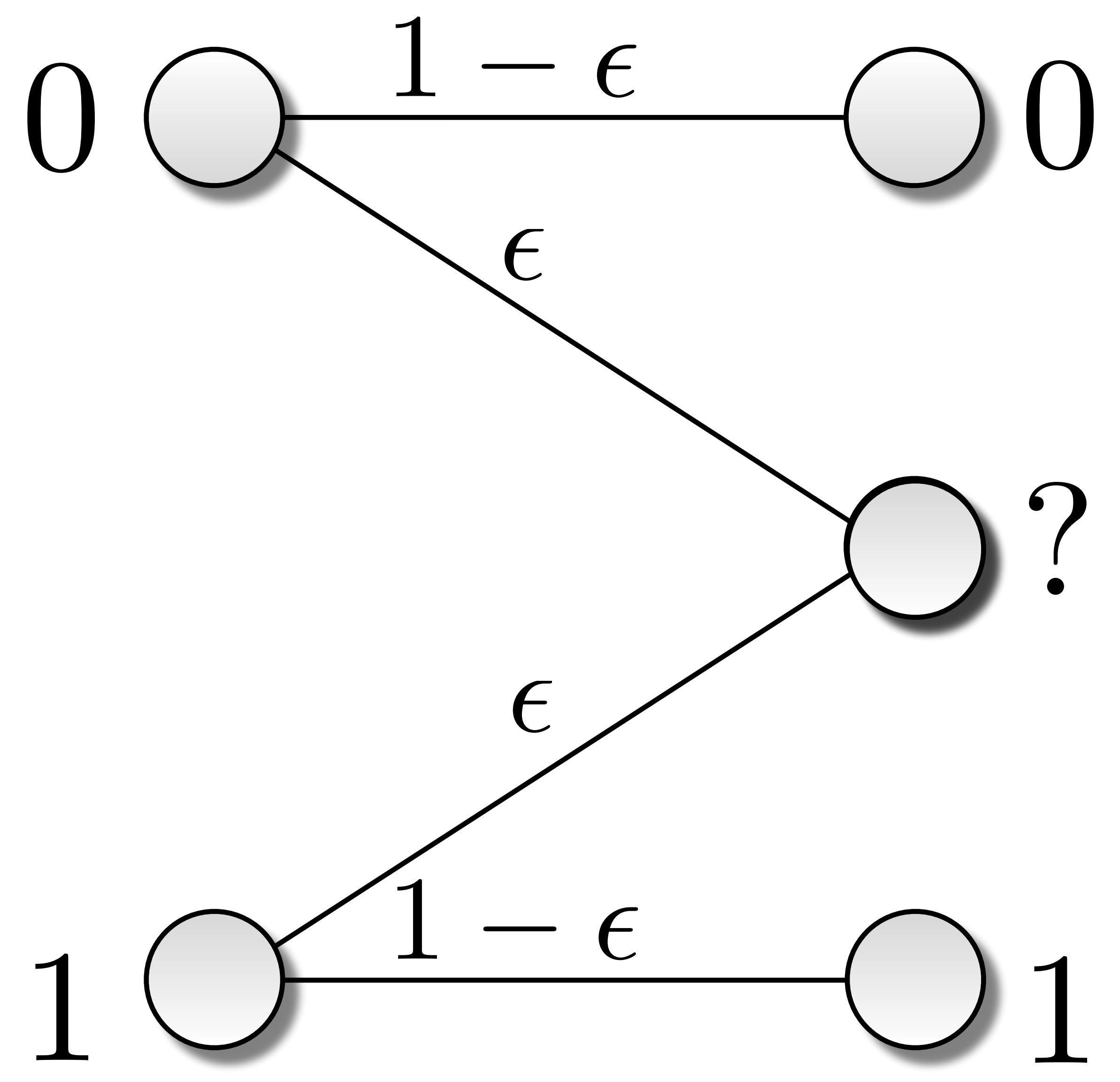
Choose LUT to maximize mutual information

$$\max I\big(X_3; Z\big) = \max_{\mathrm{LUT}} I\big(X_3; \mathrm{LUT}(L_1, L_2)\big)$$

LUT for Each Node, Each Iteration

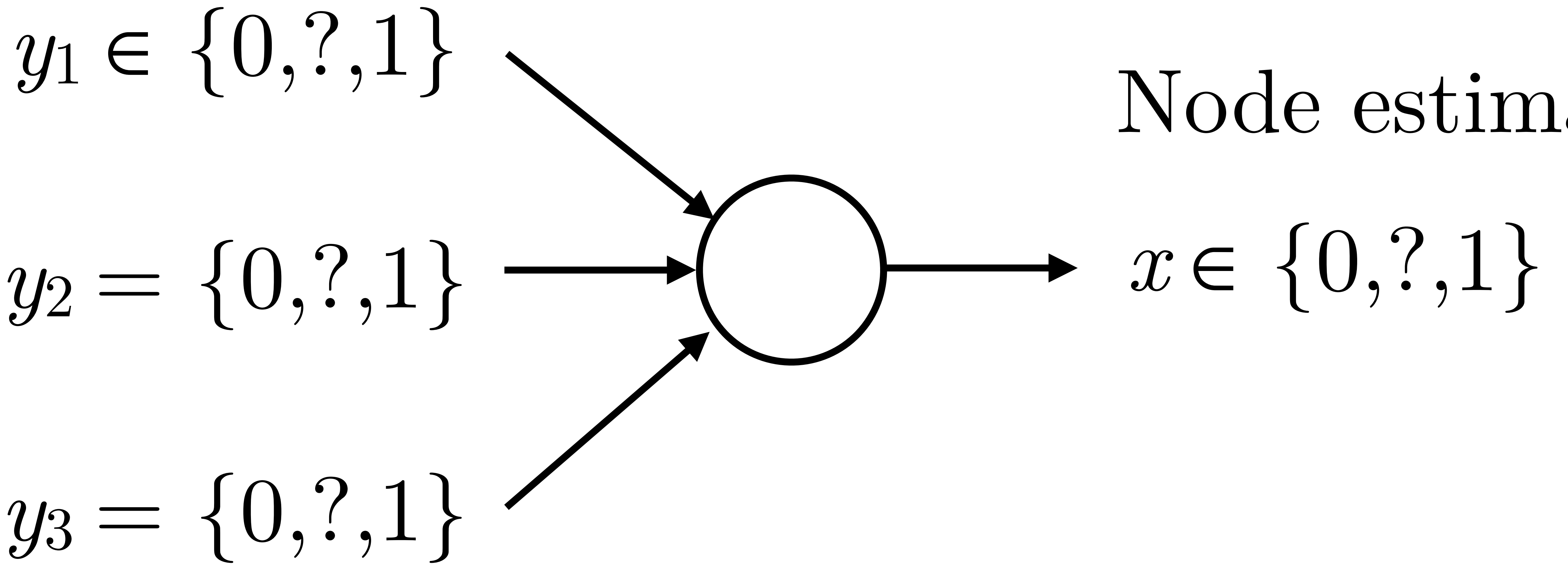


Example: Three-Level Messages



binary erasure channel (BEC)

Single bit x is transmitted over three parallel BEC's



Majority vote decoding. Examples:

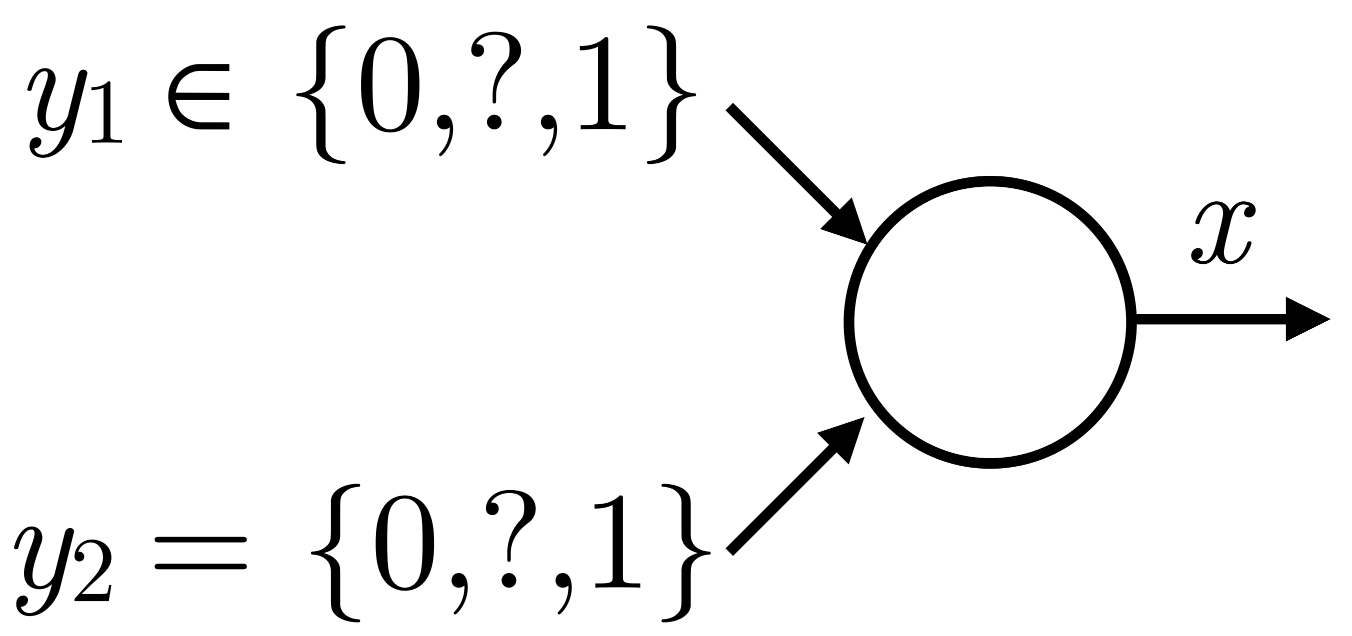
$y_1\ y_2\ y_3 = 0\ 0\ 1 \quad \rightarrow \quad x = 0$

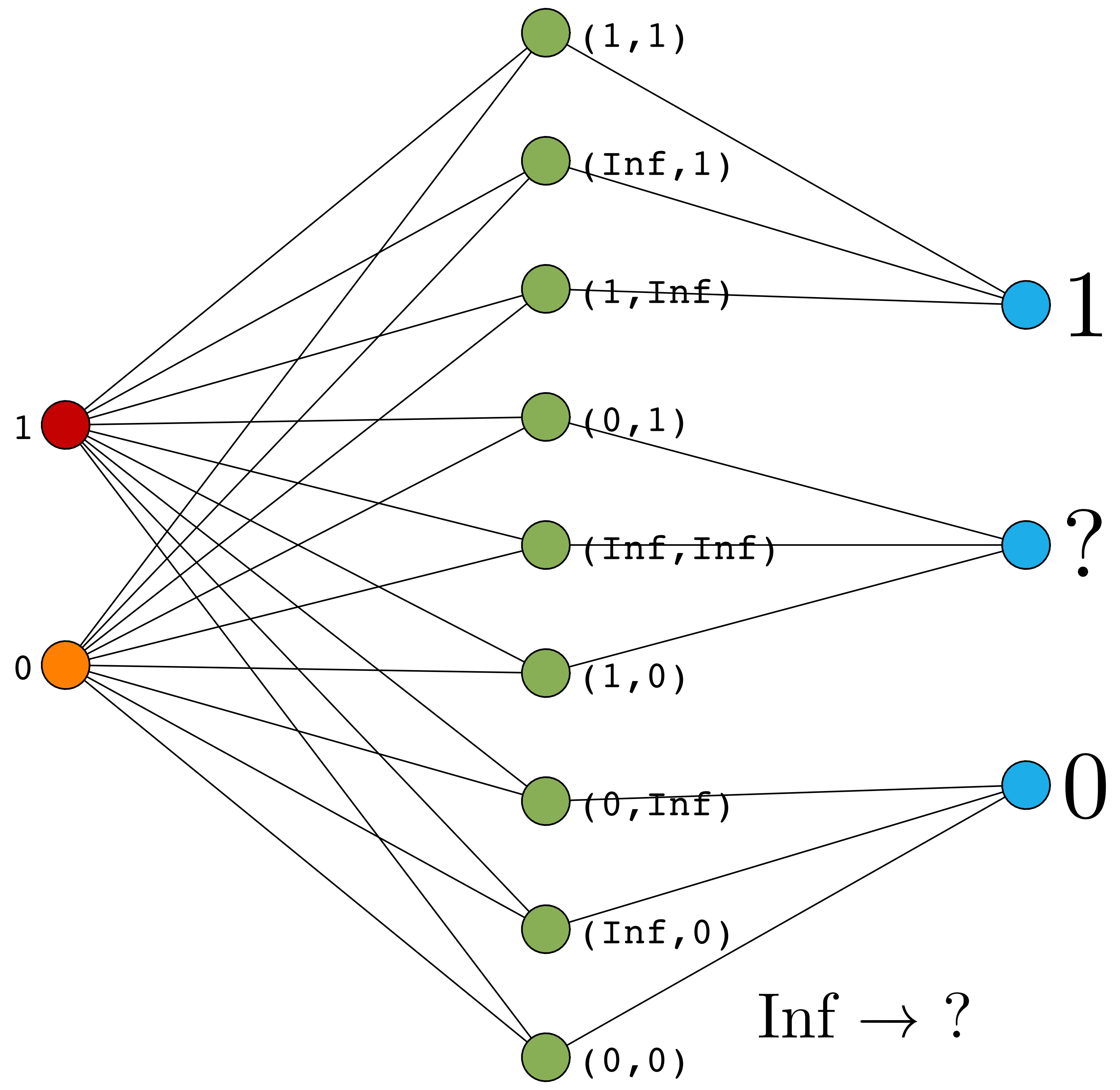
$y_1\ y_2\ y_3 = ?\ 0\ 1 \quad \rightarrow \quad x = ?$

$y_1\ y_2\ y_3 = 1\ 0\ 0 \quad \rightarrow \quad x =$

$y_1\ y_2\ y_3 = ?\ 1\ ? \quad \rightarrow \quad x =$

Max-LUT Method Designs Correct Table





y_2	$y_1 = 0$	$y_1 = ?$	$y_1 = 1$
0	0	0	?
?	0	?	1
1	?	1	1

The Max-LUT method designs the table above. It gives the "majority vote" rule.

Remarkably, machine learning finds optimal decoding rule without a priori knowledge

Machine-designed decoding rules can replace human-designed rules.

This example can be generalized to more levels, and arbitrary node types.

4 bits/message close to BP

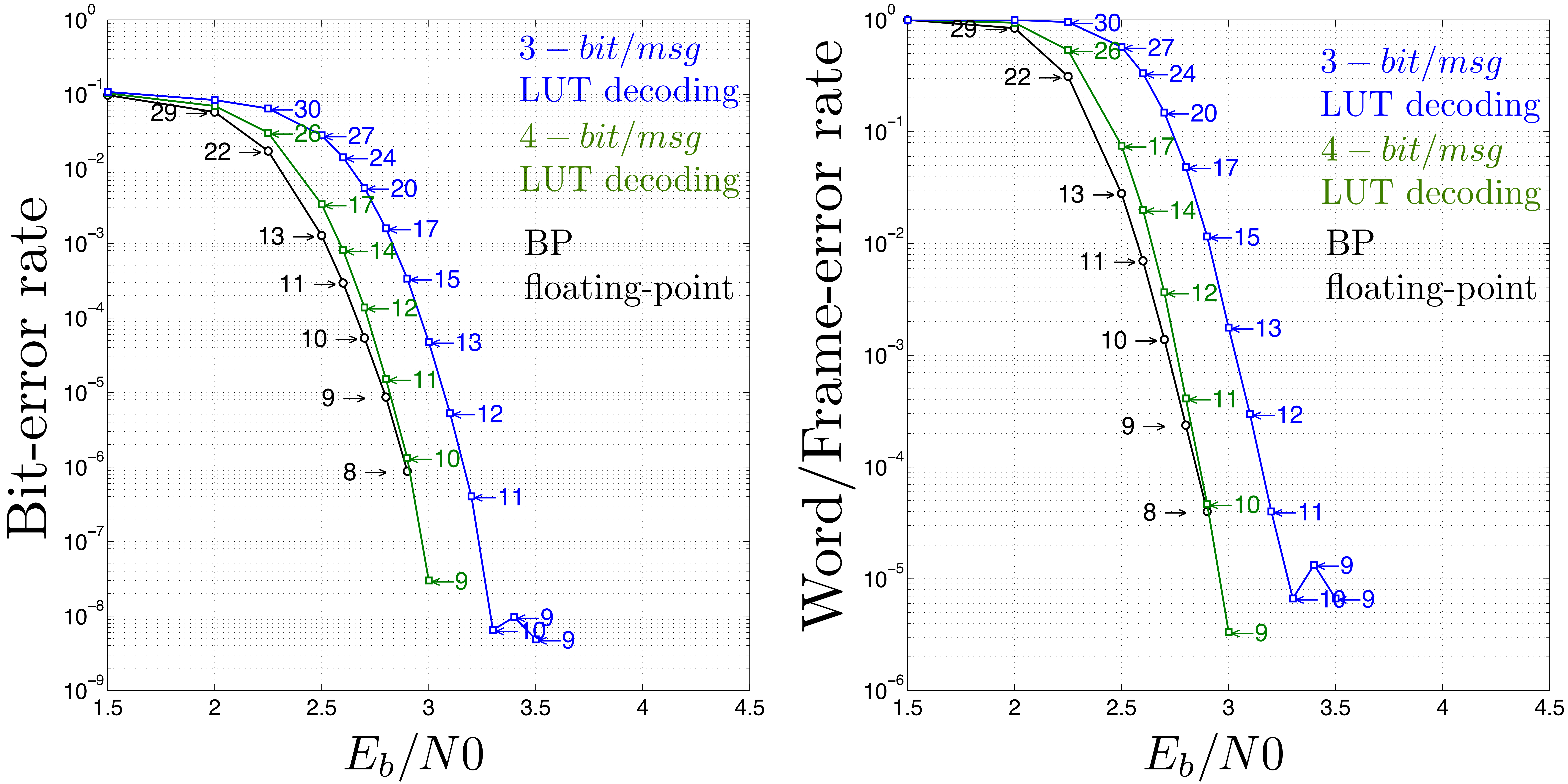


FIGURE 8. BER and WER results for the LUT decoding algorithm. $d_v = 4$, $d_c = 9$, $R = 0.56$, $N = 4113$, Max. Iter.$= 30$, Array code [2].

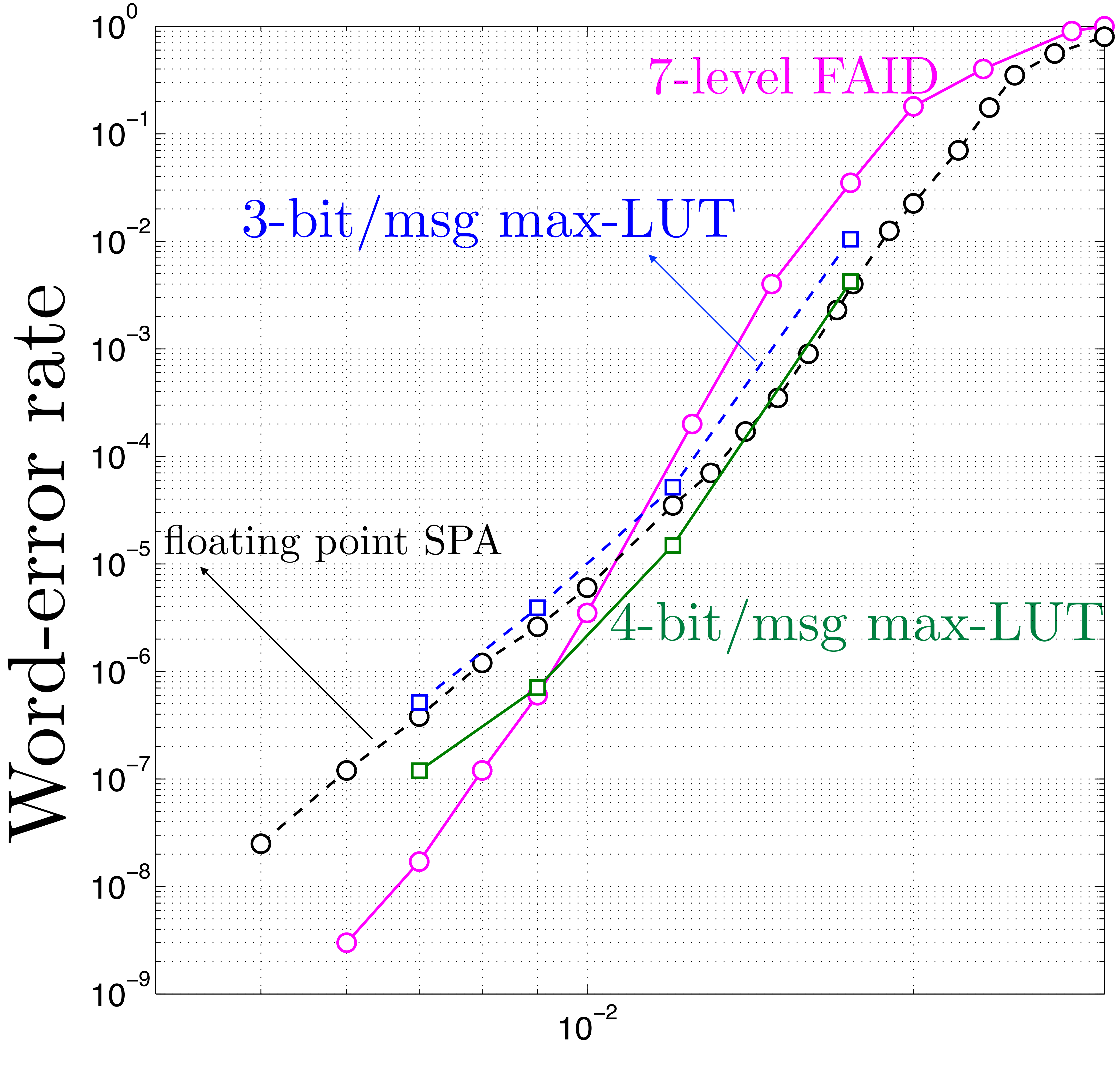
F. J. Cuadros Romero and B. M. Kurkoski, “LDPC decoding mappings that maximize mutual information,” IEEE Journal on Selected Areas in Communications, vol. 34, pp. 2391-2401, August 2016

BSC: Lower Error Floor than Sum-Product But not lower than FAIDs

$$N = 2388, (d_v = 3, d_c = 12), R = 0.75 \text{ and Max. iter} = 60$$

FAIDs are designed to avoid the effects of harmful subgraphs, lowering the error floor Planjery et al (2013).

The proposed decoding mapping functions can be used in a variety of channels not only in the BSC.



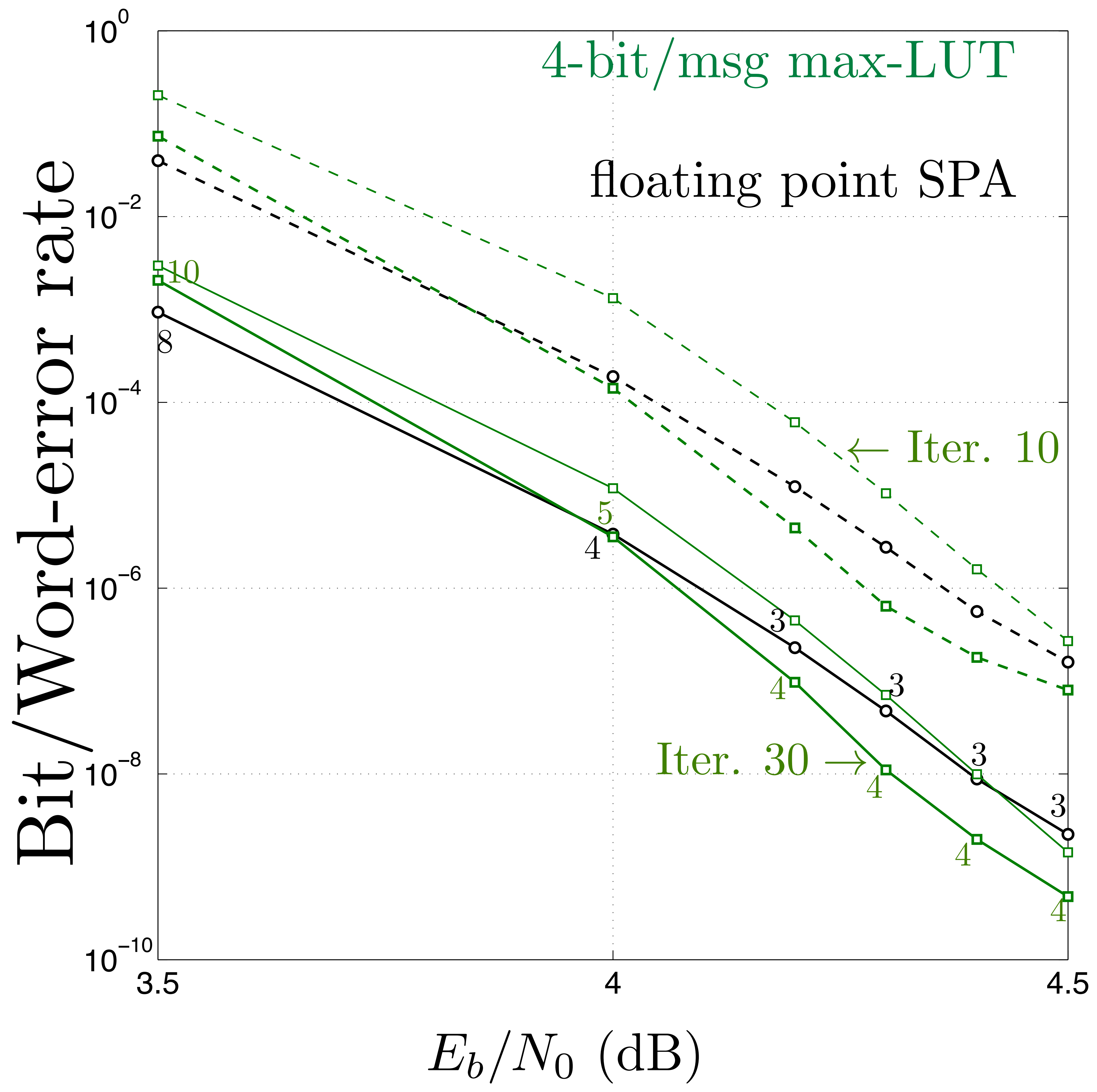
BI-AWGN: Lower Error Floors, Fewer Iterations

$N = 2048, (d_v = 6, d_c = 32), R = 0.84$ and Max. iter $= 30$

This code is used in IEEE 802.3an 10GBase-T standard producing an operation of 10 Gb/s.

The proposed decoding mapping functions

- using 10 iterations can achieve the same BER performance than full SPA using 30 iterations.
- using 30 iterations can surpass the BER performance of full SPA using 30 iterations.



Conclusion

Now is an exciting time for the application of machine learning to communications.

I showed two machine learning methods for three problems:

- Soft-input decoding of BCH codes using deep neural network
- Quantization of channels using K-means algorithm
- Improved LDPC decoding using machine learning

There are many more methods and many more problems.

To learn more...

Resources: Machine Learning for Communications

Mailing List

HDPC Mailing List: Recent papers on machine learning for communications

http://bit.ly/hdpcML → Join or leave the list

Data Sets

Collection of data sets for data-driven machine learning for communications

https://mlc.committees.comsoc.org/datasets/

Books

I. Goodfellow and Y. Bengio and A. Courville, *Deep Learning*, 2016

C. Bishop, *Pattern Recognition and Machine Learning*, 2006