Intro to machine learning (from a complex systems perspective

CSCS 530 - Marisa Eisenberg

Outline

- Basics of ML/AI
- How does this relate to complex systems?
- Unsupervised learning example: clustering
- Supervised learning example: classification
- Neural networks
- Deep learning, LLMs/transformers, etc.

What are machine learning (ML) and artificial intelligence (AI)?

- Machine learning (ML): a statistical algorithm or method that performs a task by “learning” from data

- Artificial intelligence (AI): a broader class of methods that enable computers to mimic human intelligence (e.g. vision, being able to understand and respond to language, classify objects, etc.)

- Both terms often used interchangeably, but ML is one type of AI—AI also includes deep learning, robotics, natural language processing, etc. (Although some of these are/make use of ML methods too!)

Machine Learning

Unsupervised Learning

Dimensionality Reduction
- Meaningful Compression
- Structure Discovery
- Big data Visualistaion
- Feature Elicitation

Clustering
- Recommender Systems
- Targetted Marketing
- Customer Segmentation

Supervised Learning

Classification
- Image Classification
- Customer Retention
- Idenity Fraud Detection
- Diagnostics

Regression
- Advertising Popularity Prediction
- Weather Forecasting
- Population Growth Prediction
- Market Forecasting
- Estimating life expectancy

Reinforcement Learning
- Real-time decisions
- Game AI
- Robot Navigation
- Skill Acquisition
- Learning Tasks

Types of machine learning

- **Supervised learning**: the model learns from "labeled" data—i.e. a subset of data where the answers are given
  - E.g. Classification, ChatGPT
  - Usually more focused on prediction
- **Unsupervised learning**: the model uses unlabeled data to make decisions/understand the structure of the data
  - E.g. Clustering, PCA (principal component analysis)
  - Usually less focused on prediction, more focused on understanding structure in the data

Types of machine learning

- **Reinforcement learning**: the model/system interacts with an environment or data and learns as it goes based on feedback it receives

  - E.g. Simulated Google Deepmind robots learning to walk or many iterated game theory systems

- https://www.youtube.com/watch?v=gn4nRCC9TwQ&themeRefresh=1

How does ML/AI relate to complex systems?

- Many or most ML/AI methods are based on complex systems-related tools:
    - Networks (e.g. neural networks, decision trees, random forest, causal inference methods)
    - Clustering and community detection
    - Optimization (basically all ML methods)
    - Dynamical systems (e.g. particle swarm methods etc)
    - Evolution and adaptation (genetic algorithms, adapting to new data)
    - Models and model-building

How does ML/AI relate to complex systems?

- Many complex systems researchers are ML/AI researchers and vice versa!

- But, the philosophy behind ML/AI models is often somewhat different from complex systems models!

Prediction vs. explanation

- ML methods don't usually include the underlying mechanisms —they are focused on prediction rather than understanding why the system works the way it does (explanation)

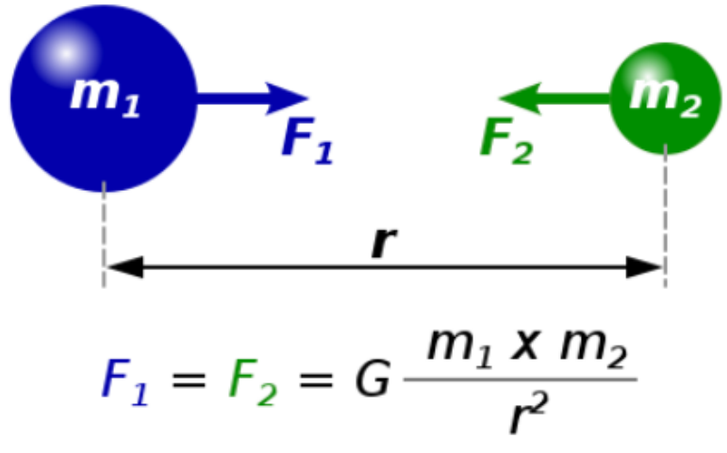
- Many ML/AI methods are "black box"—we can test their performance and see how well they work, but we don't necessarily understand how the structure of the model relates to the predictions it makes (e.g. why does a neural network do what it does? How do the weights correspond to a decision?)

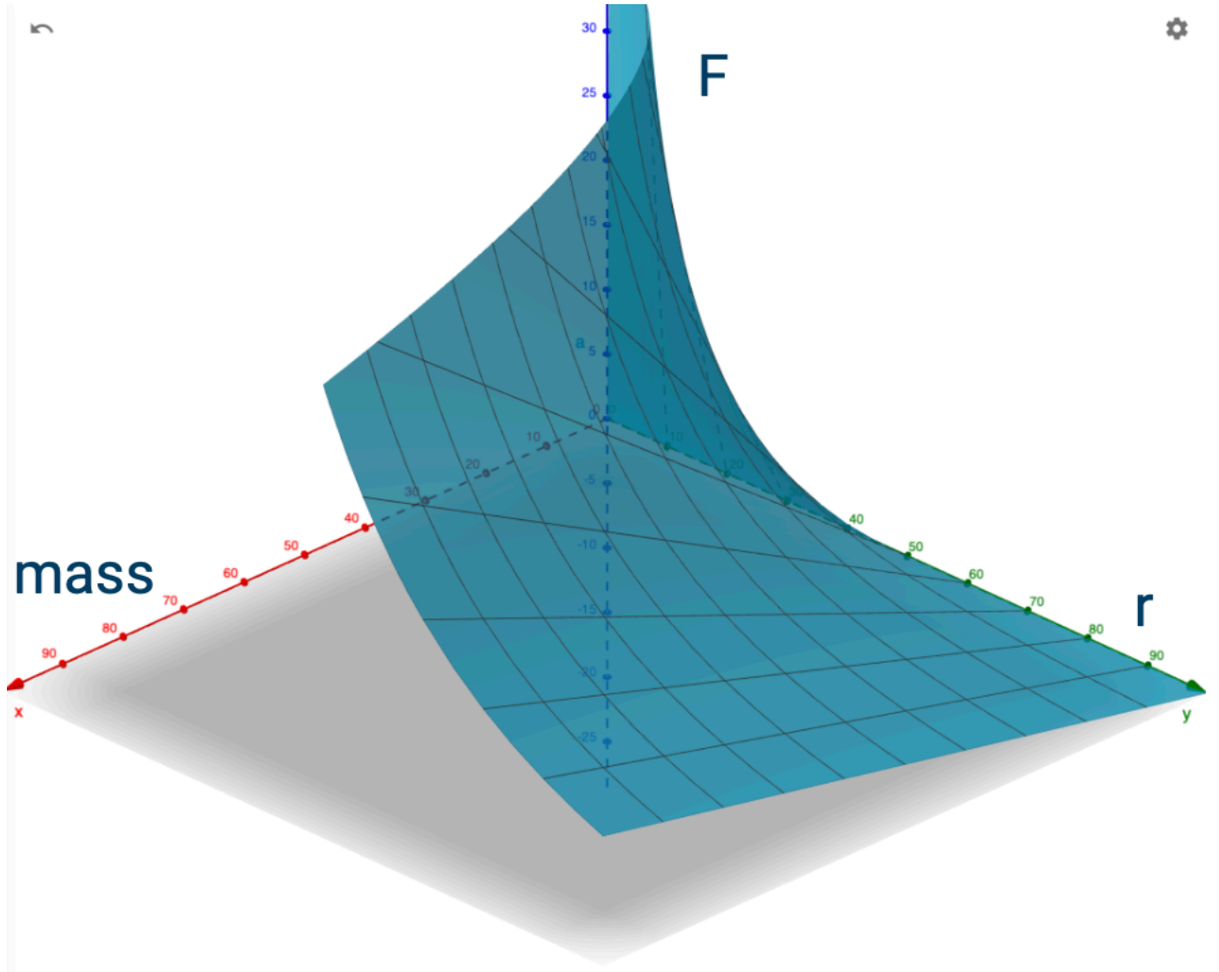
- Compare to complex systems models (e.g. Schelling model, coupled oscillators, etc., or even more standard linear models)

Do machine learning models include mechanism?

- A machine learning model like a linear model or a neural network could “learn” the gravity rule if given enough data on masses and distances

- But—what about if we were on Mars?





Why is mechanism important?

- ML/AI models don’t have to learn the correct mechanism from the data? Sometimes they learn confounding variables, or irrelevant things that relate to the training data they are given

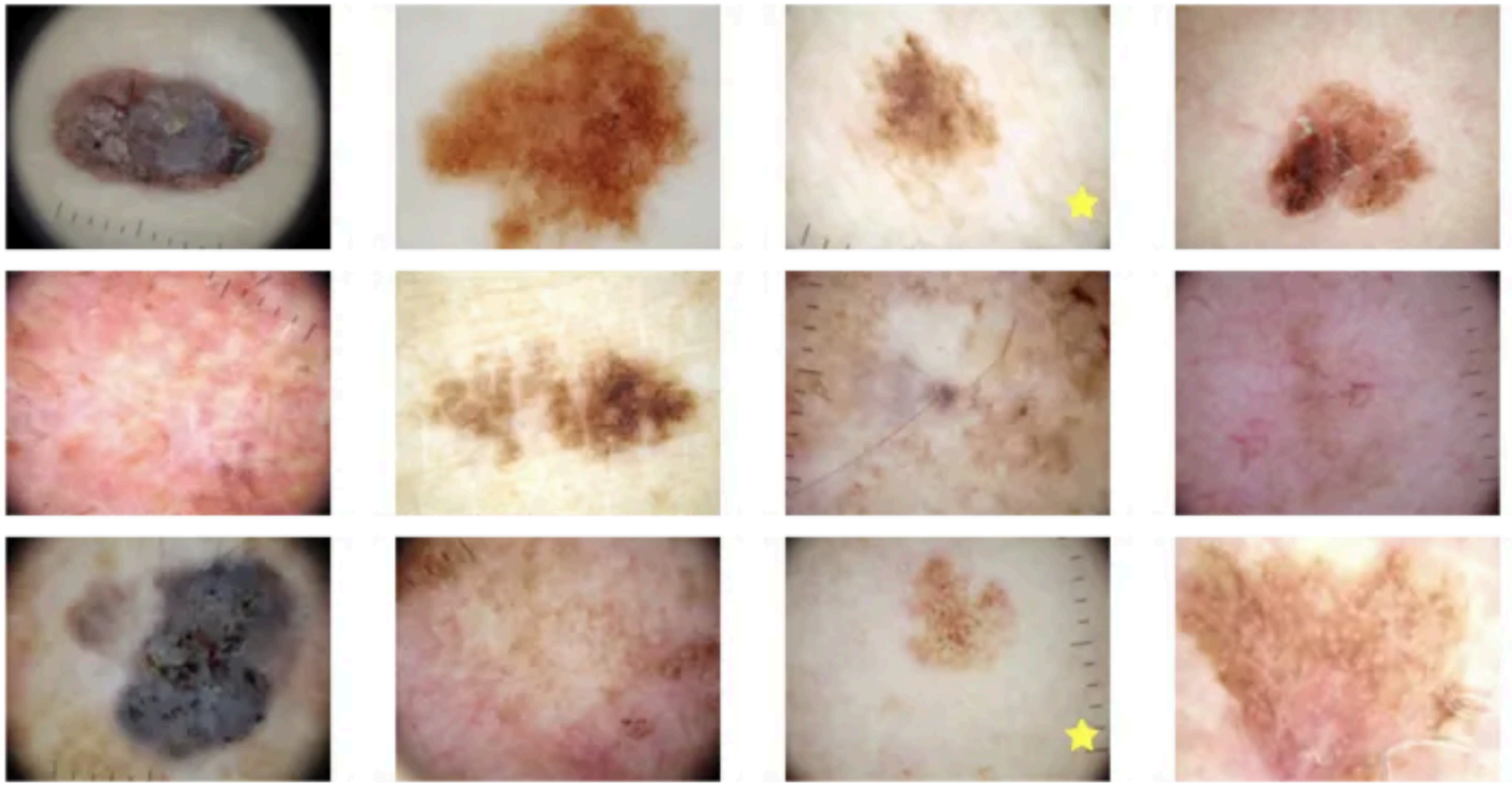
- “Clever Hans” predictors

Detecting skin cancer with AI/ML

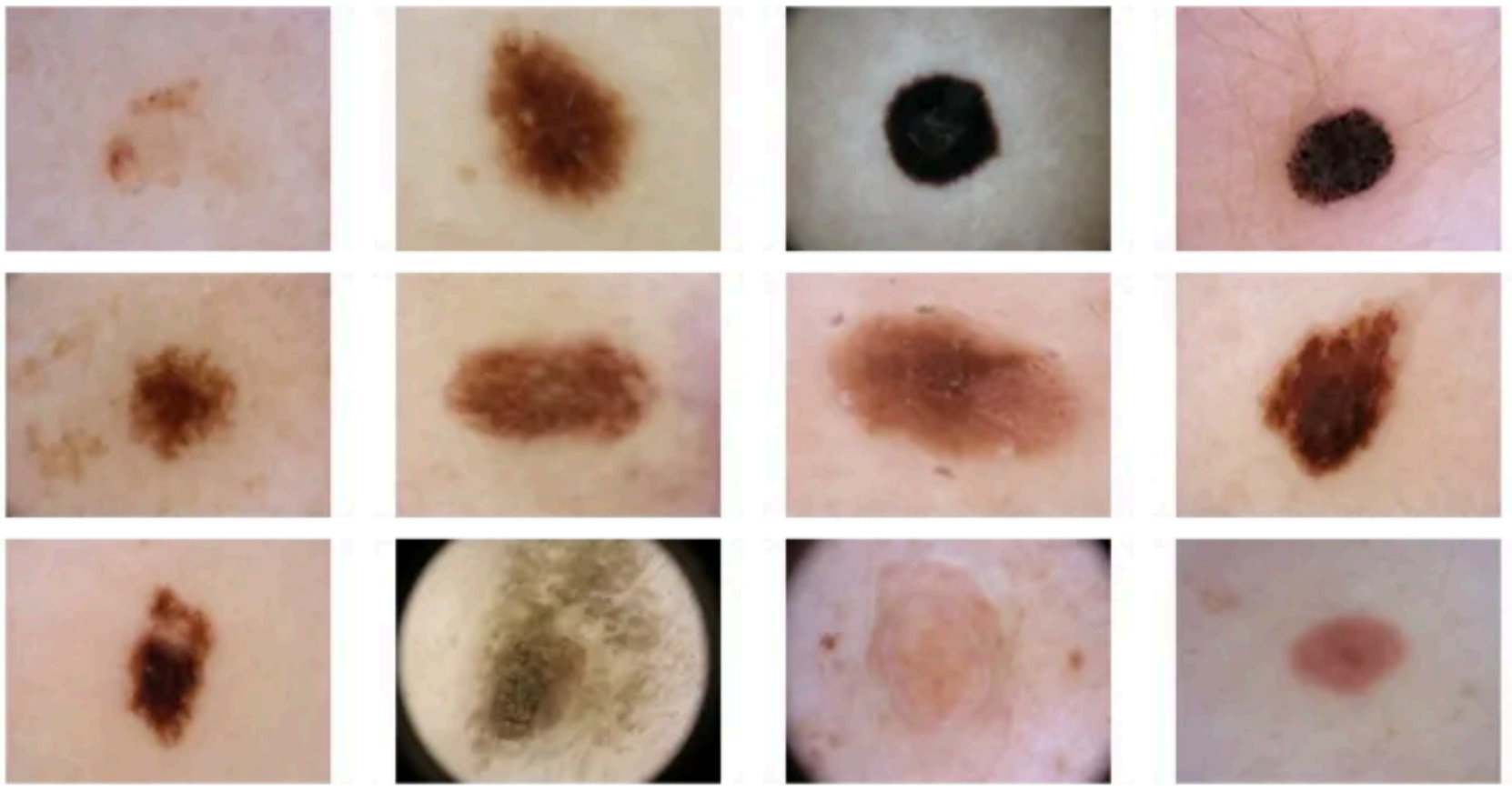
- Neural network for skin lesion classification - Nature 2017 paper

- Built on large datasets of clinical and nonclinical images of benign and malignant lesions—performed equivalently to panel of dermatologists

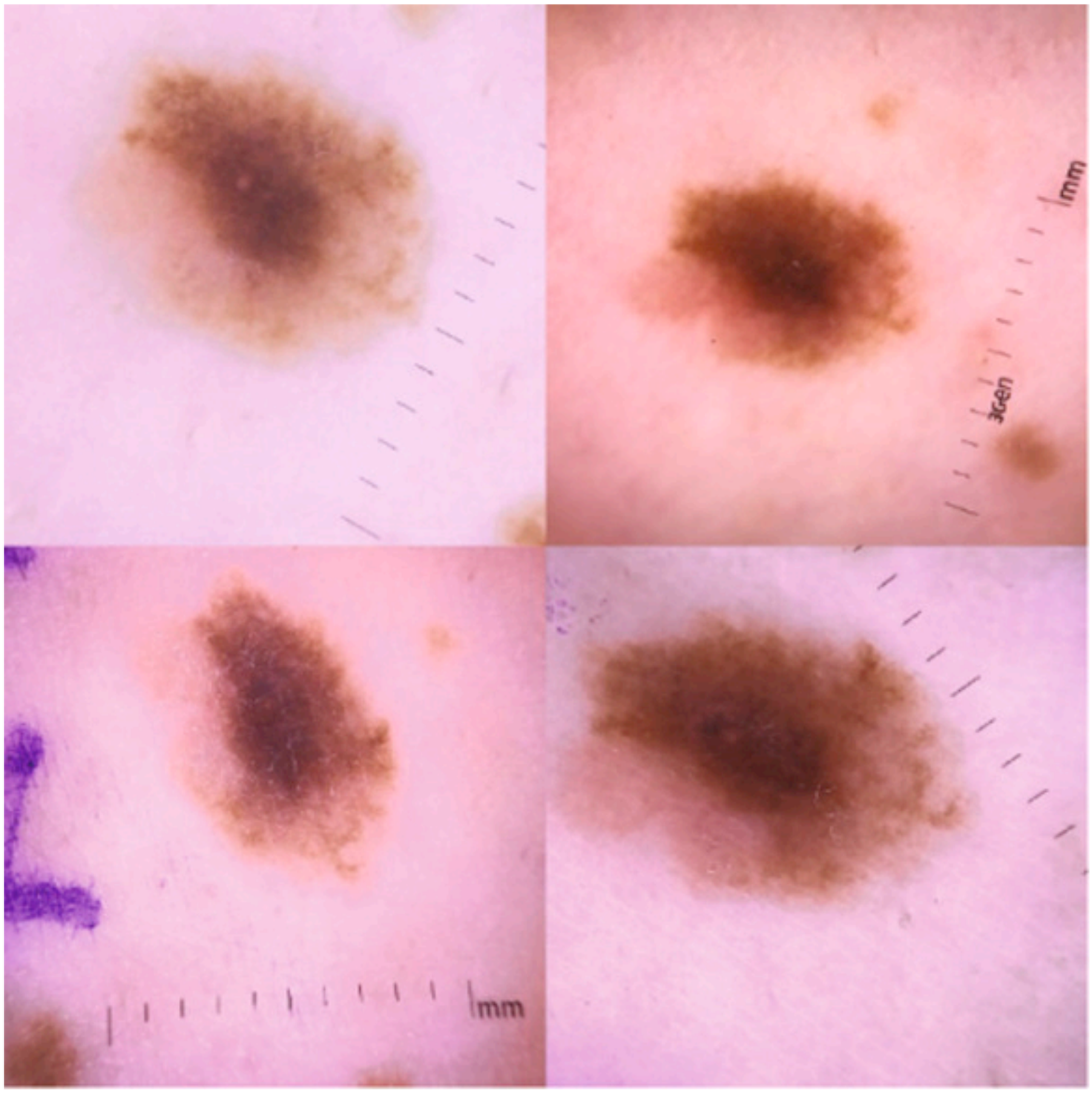
- But when tested on real data, performed worse—why?



A sample of images of **malignant cancers** used for tra
www.isic-arch



A sample of images of **benign lesions** used for training and testing the algorithm — available from www.isic-archive.com



Automated Classification of Skin Lesions: From Pixels to Practice, 2018 https://www.sciencedirect.com/science/article/pii/S0022202X18322930?via%3Dihub=

Detecting skin cancer with AI/ML

- Without mechanism, we must be extremely careful how we set up the training data and optimization (cost function etc), otherwise the algorithm can learn the wrong thing!

AI for pneumonia treatment decisions

- Caruana et al. 2015: AI system to predict likelihood of death in patients with pneumonia—to assist clinicians in deciding whether a certain person should be admitted or treated as an outpatient

- AI predicted that patients with asthma are at lower risk of severe pneumonia than those without asthma

- This is known to be false! And could result in very problematic or fatal consequences if followed without thinking

- Why?

Artificial intelligence in medicine and the disclosure of risks, 2020. AI Soc. 2021; 36(3): 705–713.

AI for pneumonia treatment decisions

- Asthmatic patients are high priority for care, and are more likely to seek care as soon as they show symptoms

- Asthma does NOT cause better pneumonia outcomes, but it correlates because those who are asthmatic are treated differently and earlier

- AI doesn’t hypothesize a mechanism underlying the system

- AI learns correlations, not necessarily causation

- We have to be very careful about how we train these models —what optimization setup (e.g. cost function) and input data

Artificial intelligence in medicine and the disclosure of risks, 2020. AI Soc. 2021; 36(3): 705–713.

“Black box” AI/ML can be problematic for medical and other high-stakes decision-making

- Because of the risks of systematic bias, mismatch/ confounding/“Clever Hans” predictors, and cyber-attacks, there are calls in the medical community to disclose the use of AI to patients as part of the discussion of risk (similar to how we discuss when doing genetic screening)

- Concerns that AI can propagate biases or potentially worsen medical decision-making if not implemented very carefully

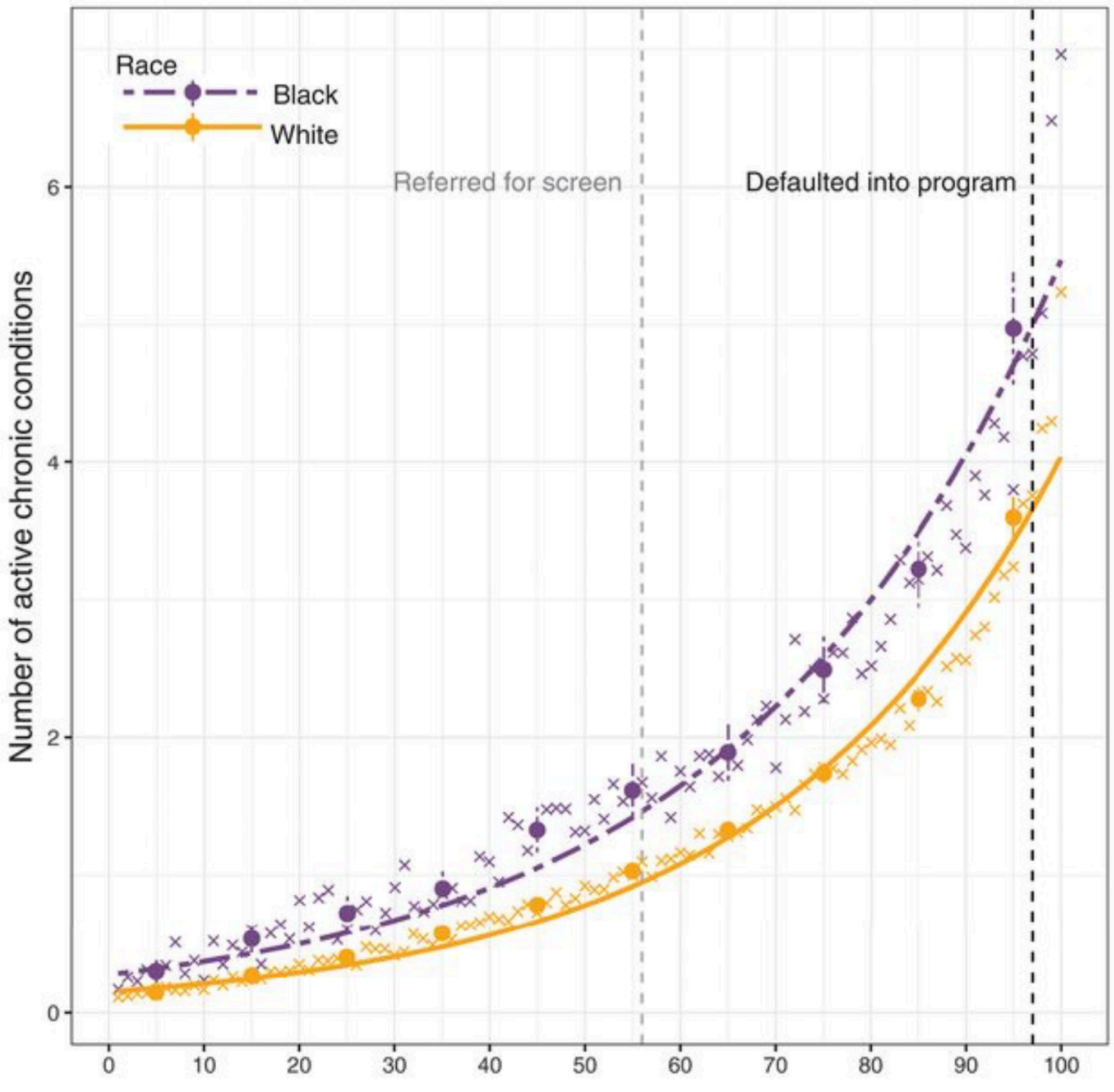
Artificial intelligence in medicine and the disclosure of risks, 2020. AI Soc. 2021; 36(3): 705–713.

Algorithmic bias, fairness, and justice

- Amazon hiring
- Healthcare decision-making
    - Health systems use commercial prediction tools (AI/ML) to identify and help patients with complex health needs
    - Obermeyer et al. (2019) showed that a widely used algorithm, affecting millions of patients, exhibits significant racial bias
    - “At a given risk score, Black patients are considerably sicker than White patients, as evidenced by signs of uncontrolled illnesses”

Obermeyer et al 2019: https://www.science.org/doi/10.1126/science.aax2342

A



https://www.science.org/doi/10.1126/science.aax2342

Algorithmic bias, fairness, and justice

- "Remedying this disparity would increase the percentage of Black patients receiving additional help from 17.7 to 46.5%"

- "The bias arises because the algorithm predicts health care costs rather than illness, but unequal access to care means that we spend less money caring for Black patients than for White patients."

- "Thus, despite health care cost appearing to be an effective proxy for health by some measures of predictive accuracy, large racial biases arise."

- Be careful about your input data, cost function, etc! Proxies like health care cost can be very problematic

Obermeyer et al 2019: https://www.science.org/doi/10.1126/science.aax2342

How can complex systems methods help us understand why ML/AI methods work and how they work?

- Complex systems tools are built to help us understand and simplify large, high dimensional systems

- ML/AI are models (and so they are complex systems methods in a sense)—but they are also themselves large, high dimensional systems that exhibit complex emergent behavior

- Indeed, the behavior we are training these models to exhibit is an emergent property of the interaction of many model components (e.g. neural networks)

- Complex systems tools can help us understand how and why they do what they do

Emergence and ML/AL

- What's the difference between a statistical model and a machine learning model?

- Scale—this is in some ways a very complex systems thing in that the target behavior of something like a neural network is really an emergent property of a large number of "neurons" interacting with each other (i.e. feeding in input from one to the other)

- Neural networks are networks, also ABMs basically

Emergency and AI/ML

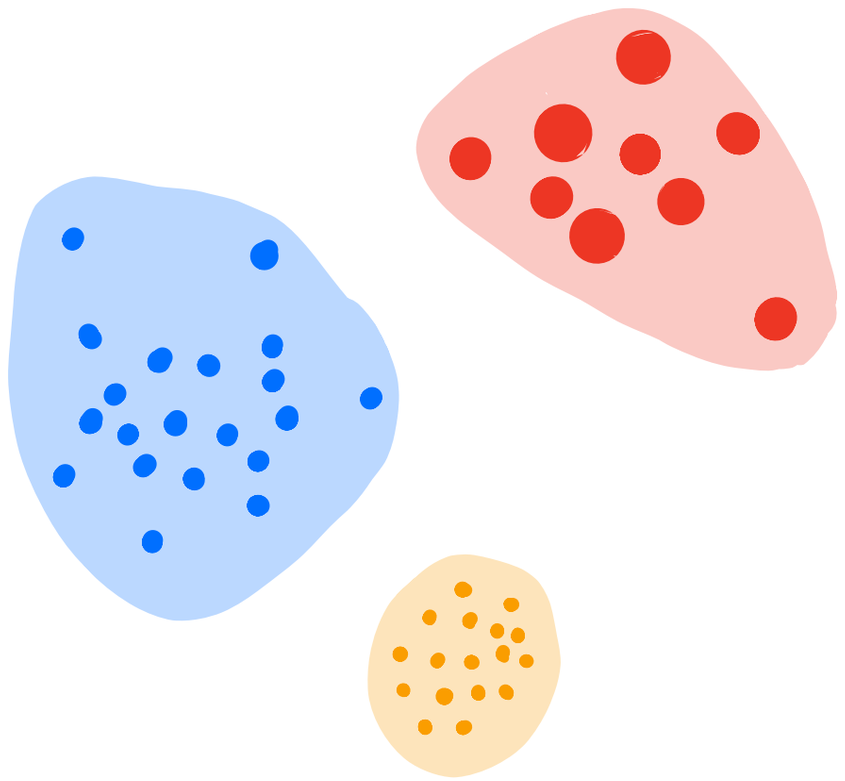
- Even though as models these are not very complex systems-y in their philosophy, as systems they are very complex systems-y

- More about this when we talk about explainable AI and mechnanistic interpretability

Unsupervised learning example: **clustering**

Cluster Analysis

- What is a cluster?
    - A set of objects/data points, such that the objects in the set are more similar to one another than they are to the objects outside the set/other clusters.



Uses of clustering in a modeling/ABM context

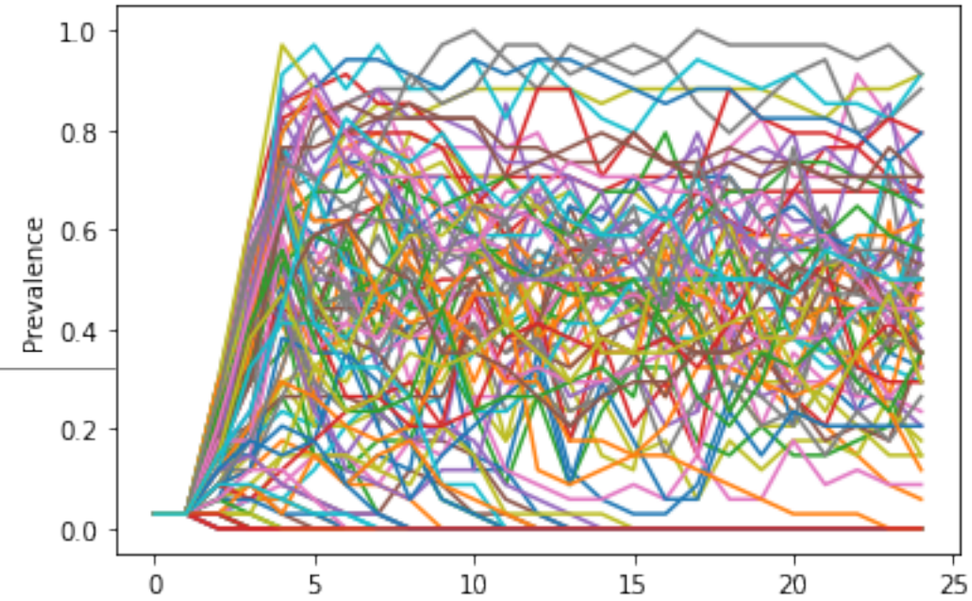
- Clustering methods can be used for a range of purposes
- Use for data analysis and understanding qualitative patterns in data set
    - Can use this to validate model outcomes when quantitative fitting is less possible
- Use clustering methods to understand the types of model behaviors observed across a sample of parameters (e.g. sample parameter space and then cluster the model results)

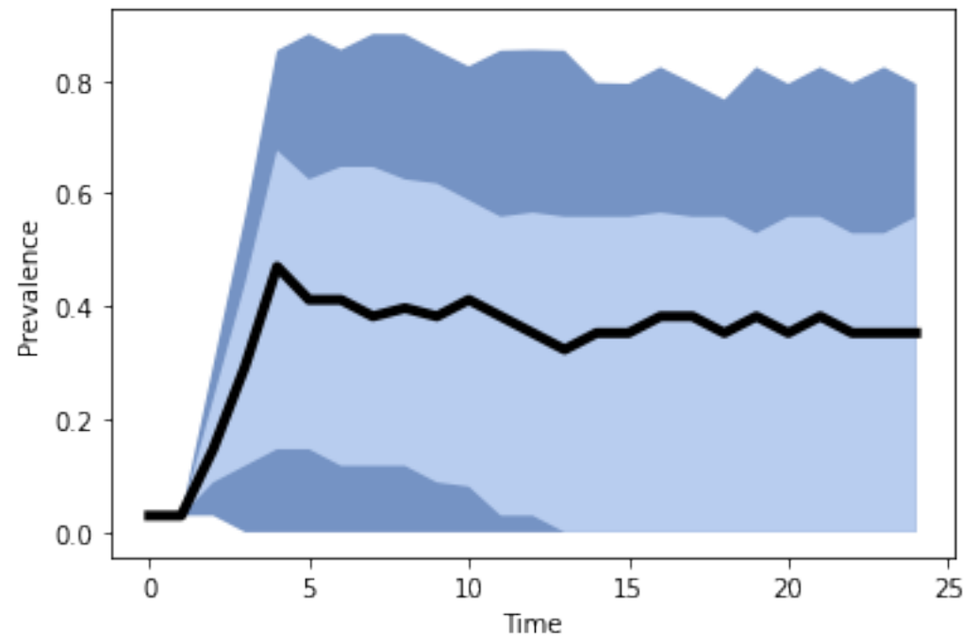
SIS network model example

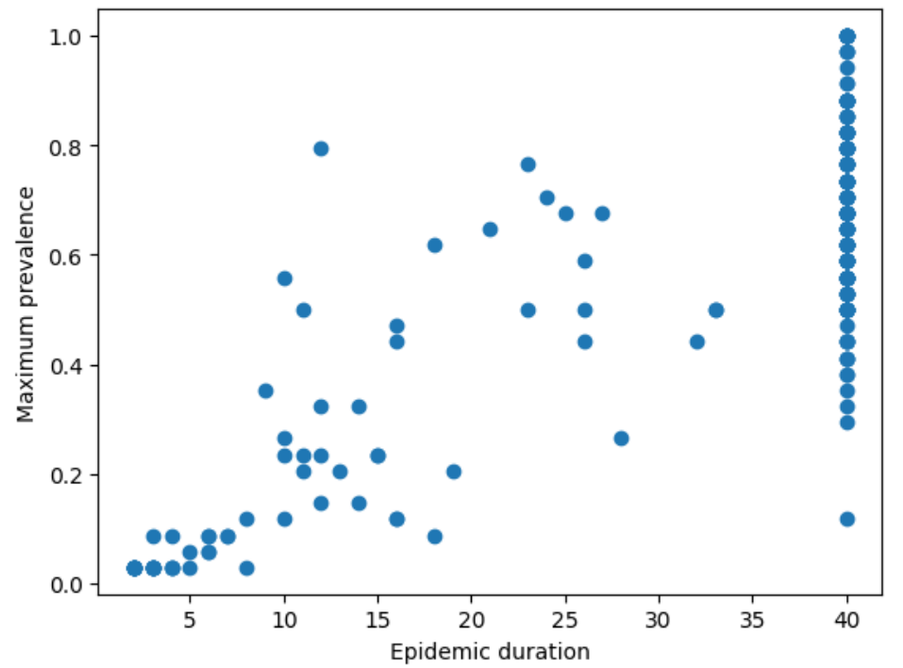
- There are qualitative patterns differences (did the epidemic die out or persist) that aren't obvious from the quantile plot

- Clustering methods can be used to uncover these patterns in an automated way (and when higher dimensional outputs)

- Could use clustering approaches on the full trajectory or summary outputs like peak prevalence, epidemic duration







Cluster Analysis

- Broadly used in data analysis, including machine learning
- **Clustering** (unsupervised) vs. **classification** (supervised)
- **Hard clustering** (every element belongs to only one cluster) vs. **fuzzy clustering** (every element has various probabilities of belonging to a given cluster)
- Some methods find the number of clusters, others use a predefined number of clusters

Cluster Analysis

- Wide range of methods—which is best depends on the data to be clustered. Not really one ‘best’ method across all settings.

- In general, we want:
  - High intra-cluster similarity, low inter-cluster similarity (how to determine similarity?)
  - Potential to discover hidden features (especially in high dimensional data)

Some general classes (or clusters haha) of clustering methods:

- **Partitioning** methods (e.g. k-means clustering & other centroid methods)
- **Hierarchical** clustering methods
- **Density-based** methods
- **Model or distribution-based** methods (e.g. Gaussian mixture models, latent class analysis)
- Network clustering methods (**community detection** methods)
- & many others!

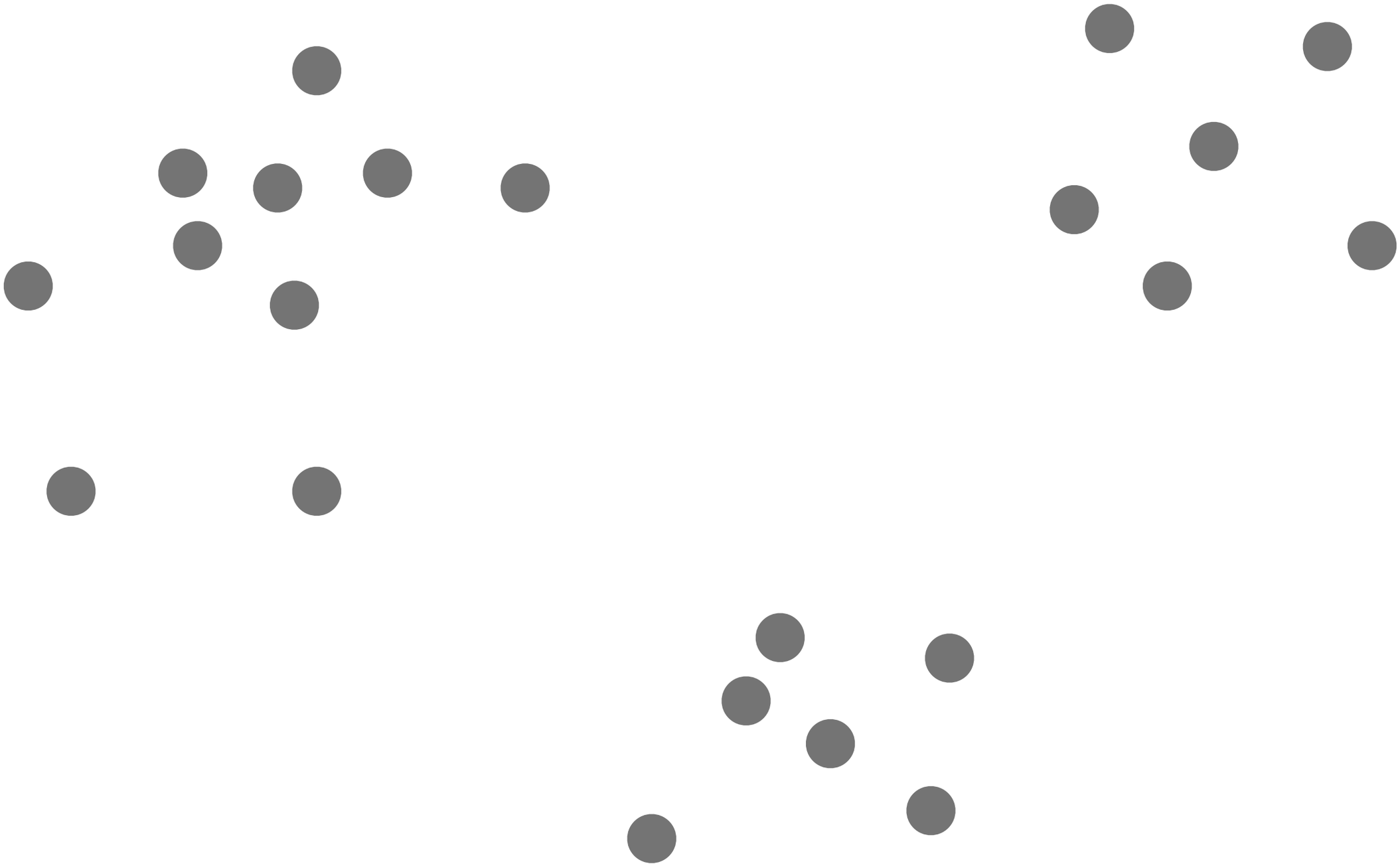
Partitioning methods

- General idea is often:
  - Construct a partition of the data into *k* clusters
  - Evaluate the resulting clusters and improve the partition
  - Repeat until optimal partition/clusters found
- Examples: k-means, k-medioids, k-modes (among many others)

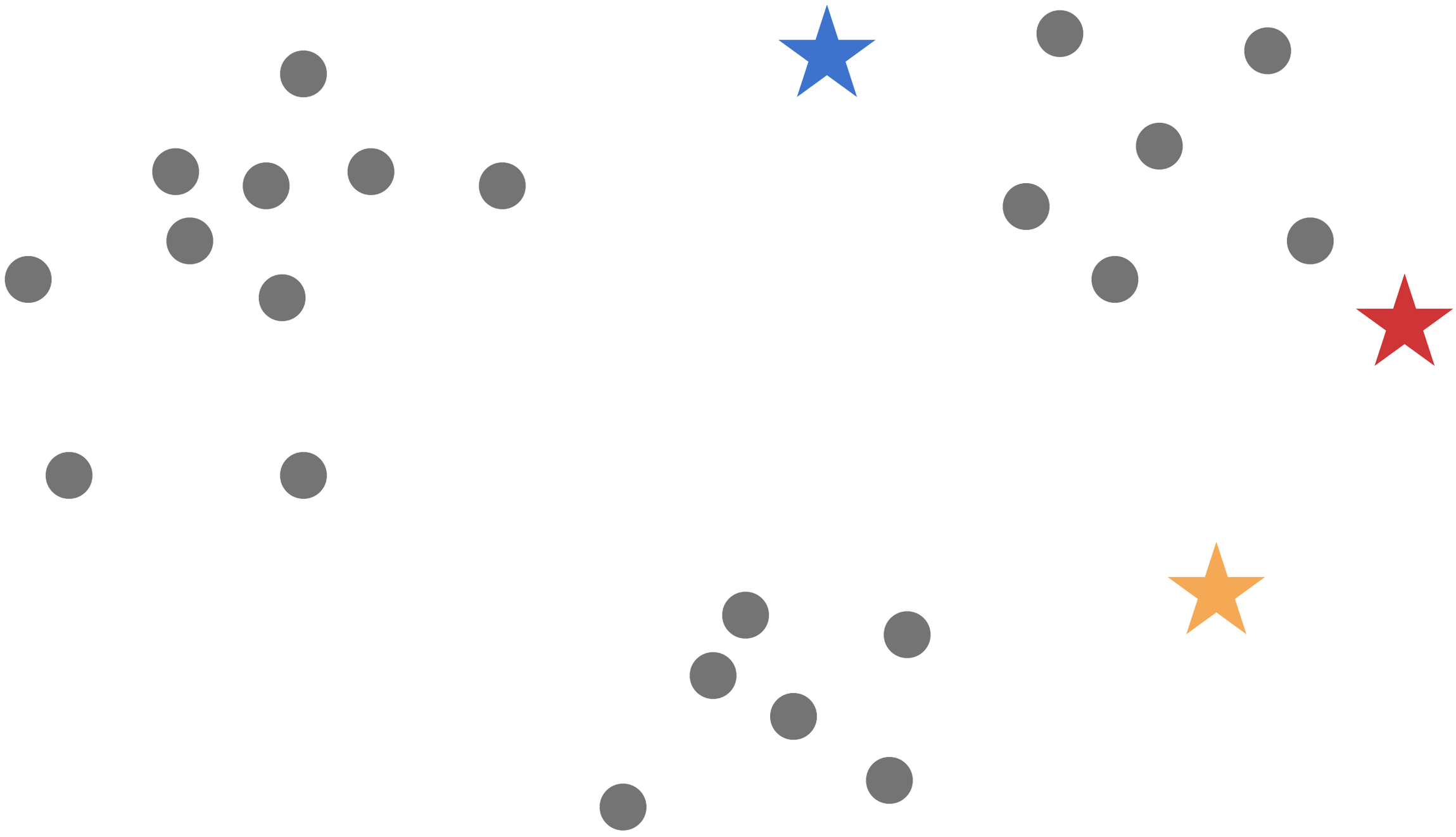
K-means clustering

- Select k centroids (means), and each data point is assigned to the nearest centroid
- This partitions the space into Voronoi cells, which are our clusters
- For each cluster, calculate the centroid of all points
- These become the new cluster centroids
- Reassign points to nearest centroid and repeat

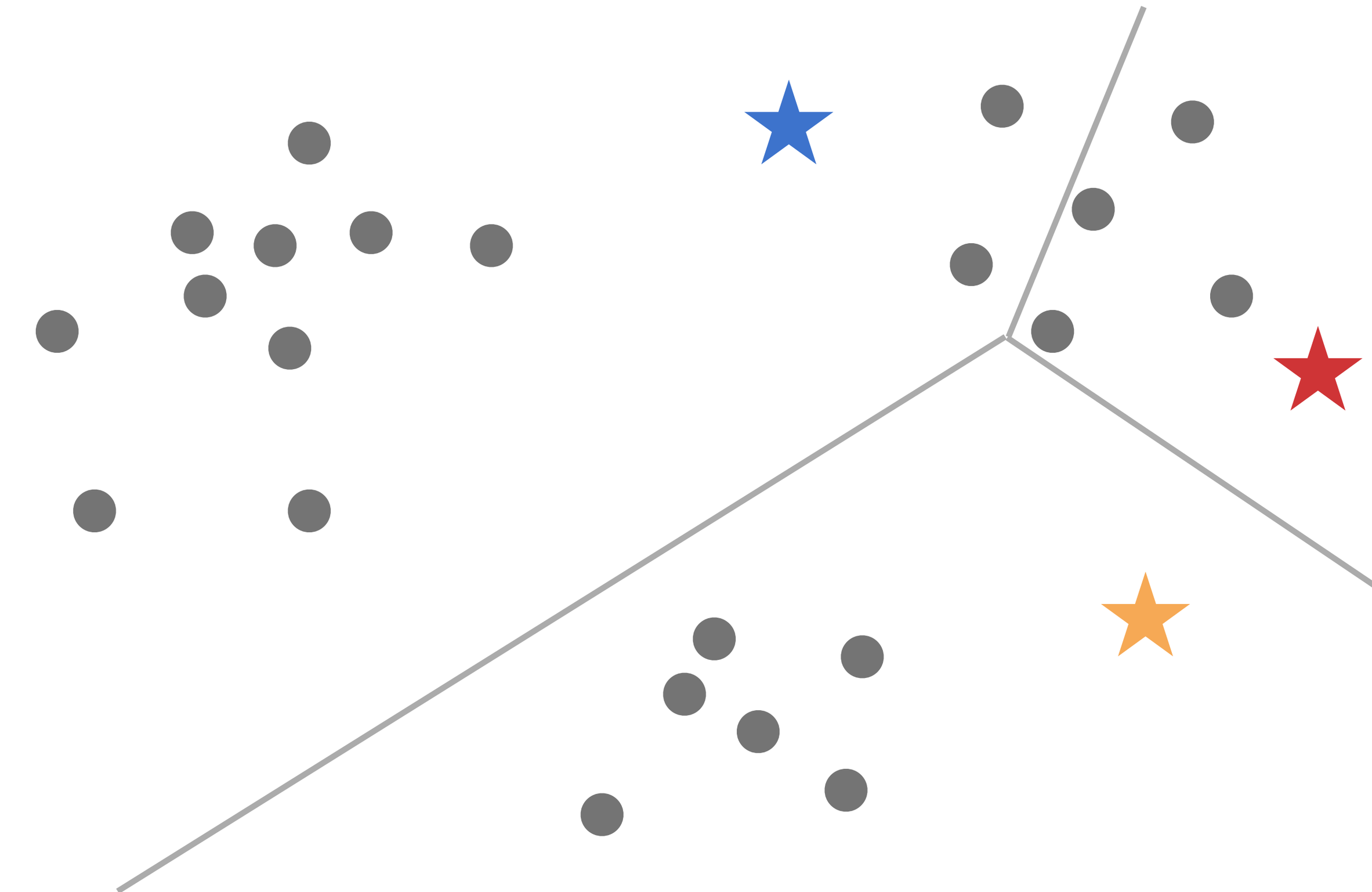
K-means clustering example



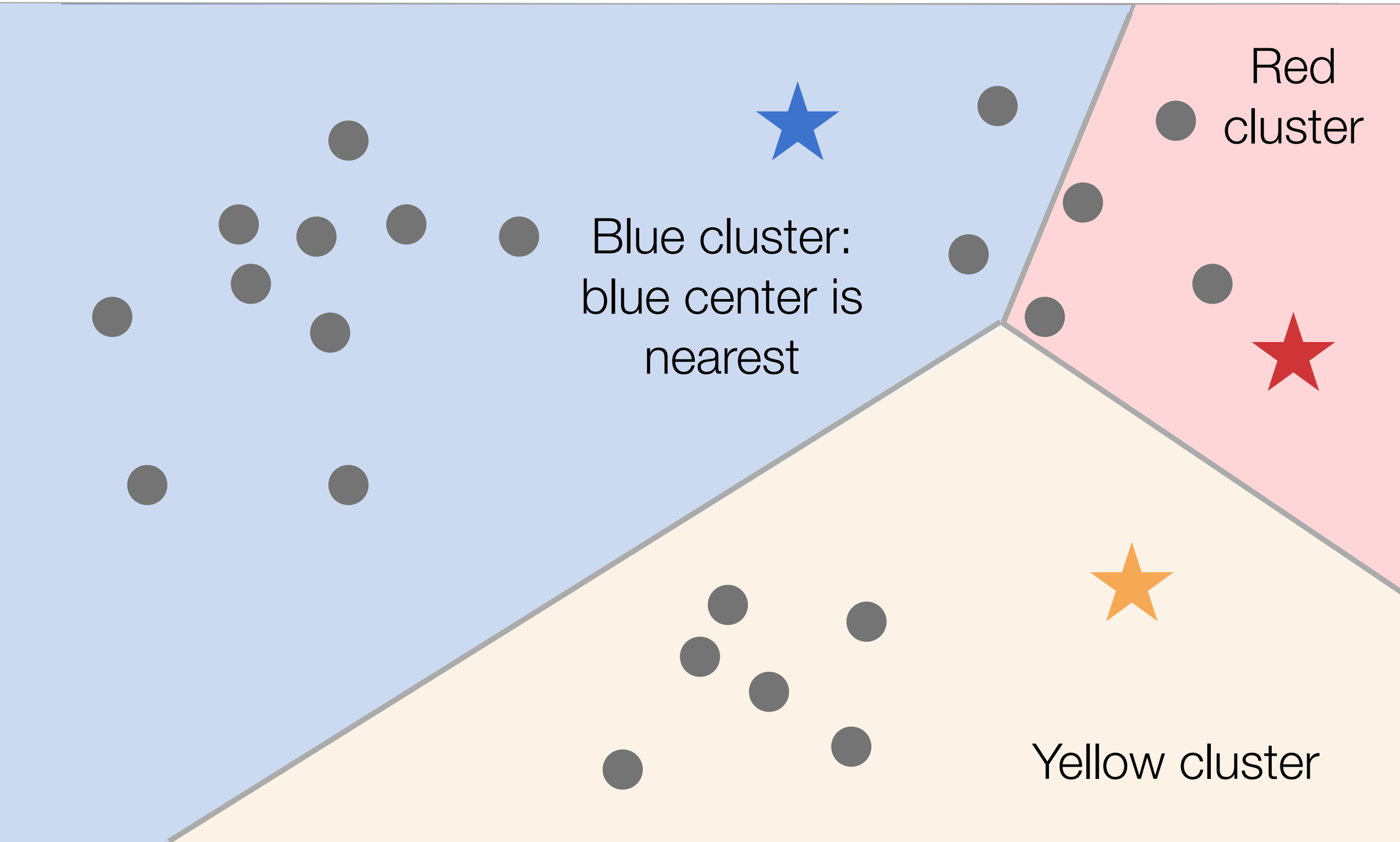
Randomly choose 3 cluster centers to start



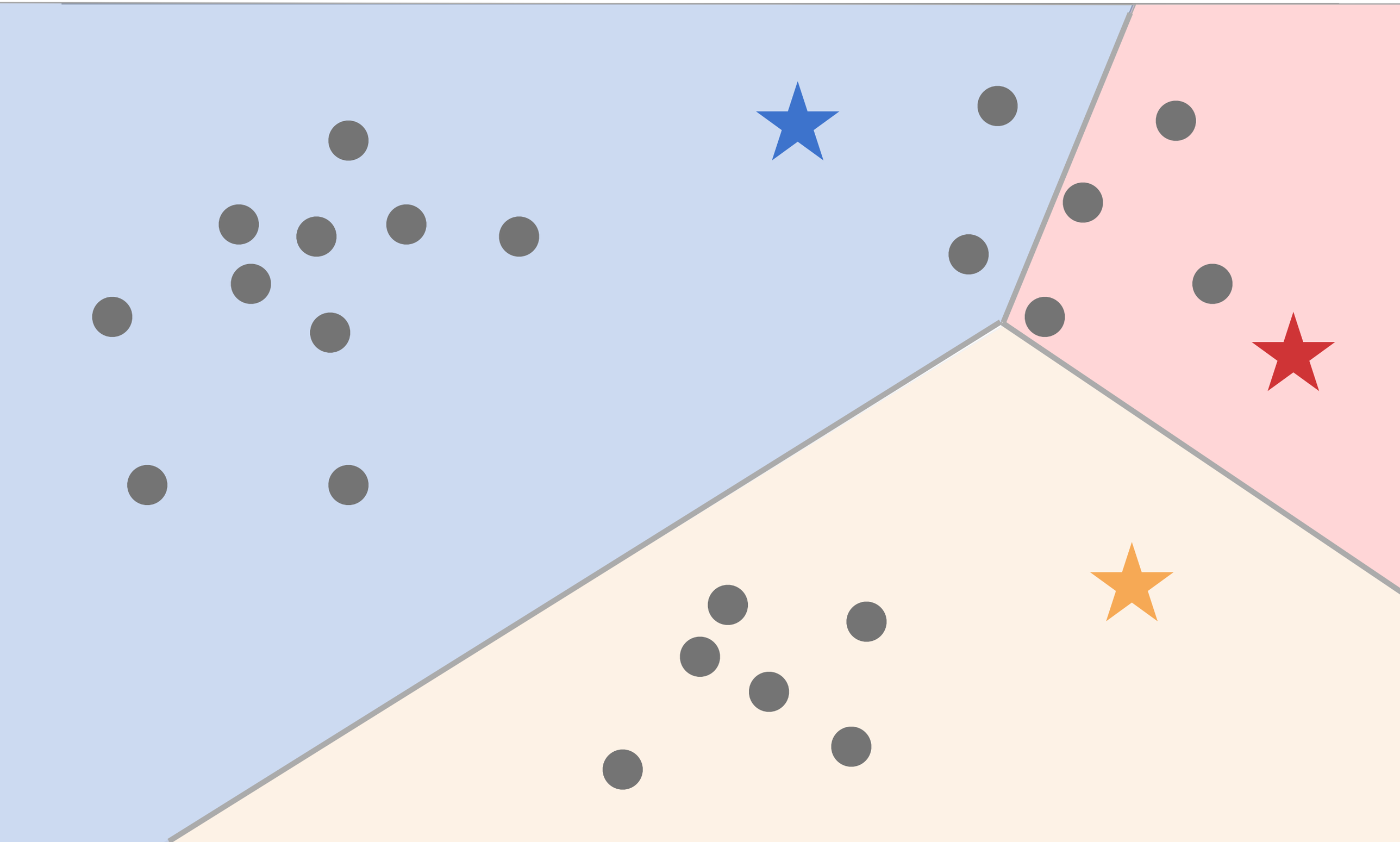
The cluster centers partition the space based on which center is nearest



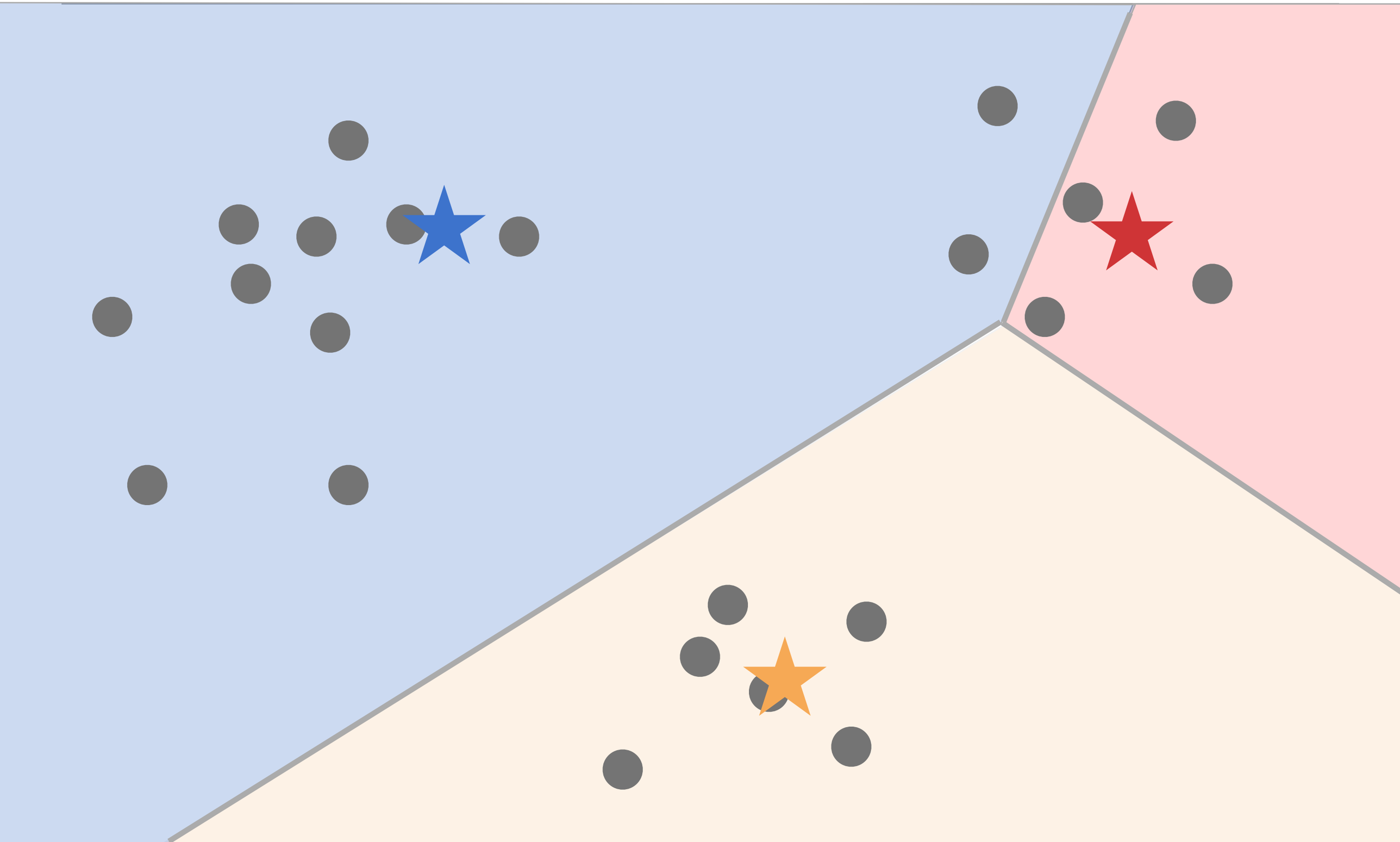
These are our starting **clusters**



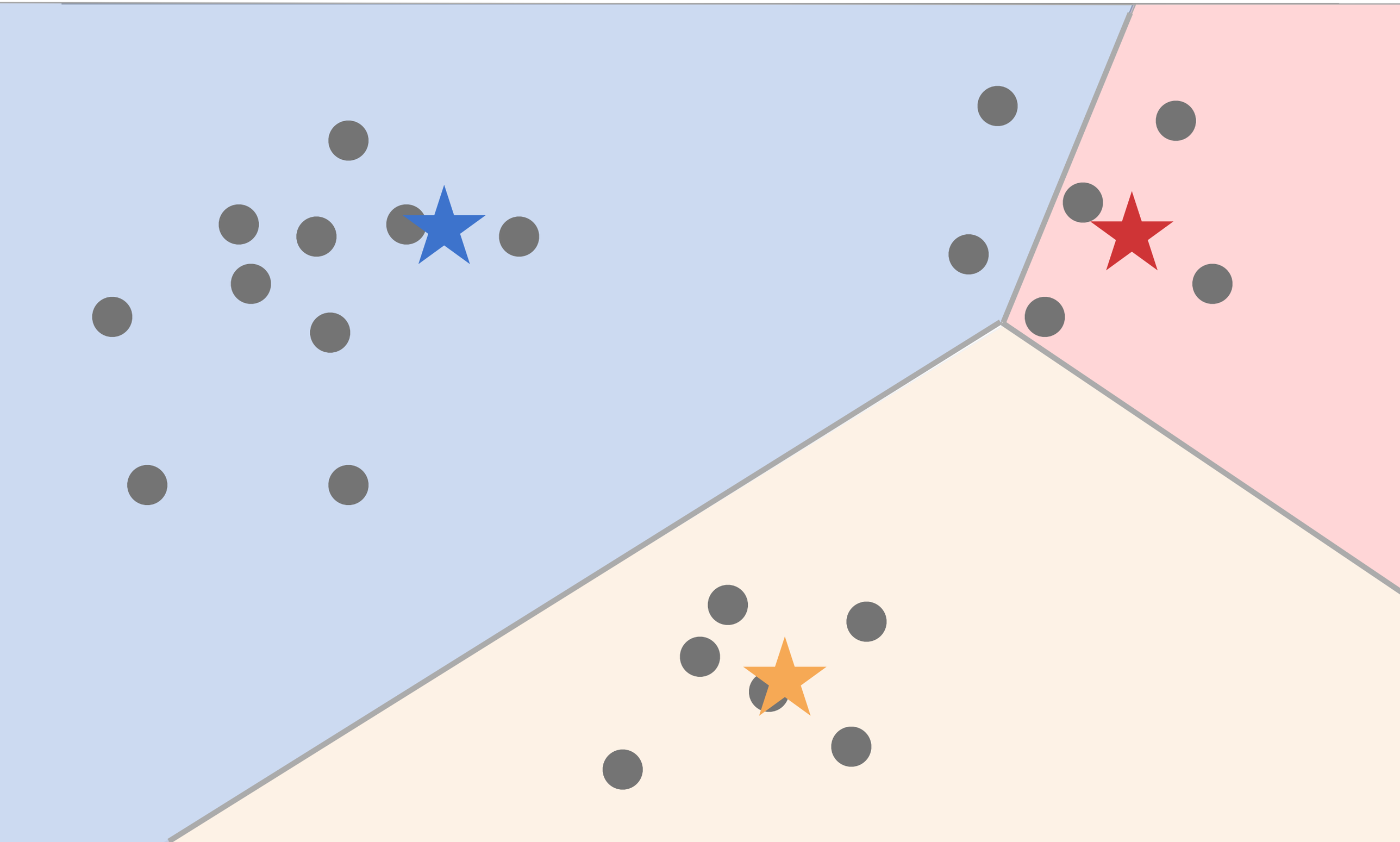
What are the means of the data points in each cluster?



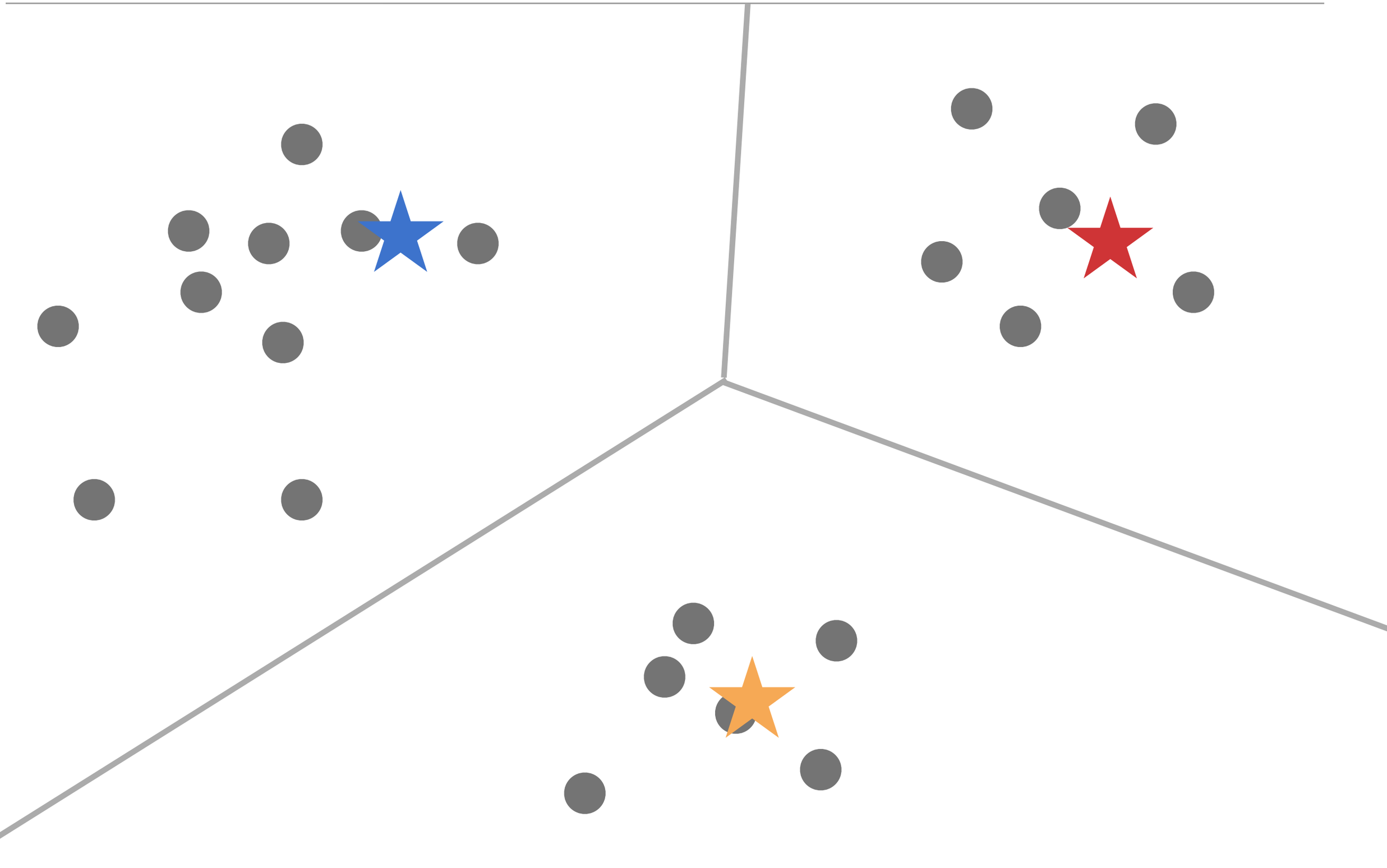
These are the new centers.



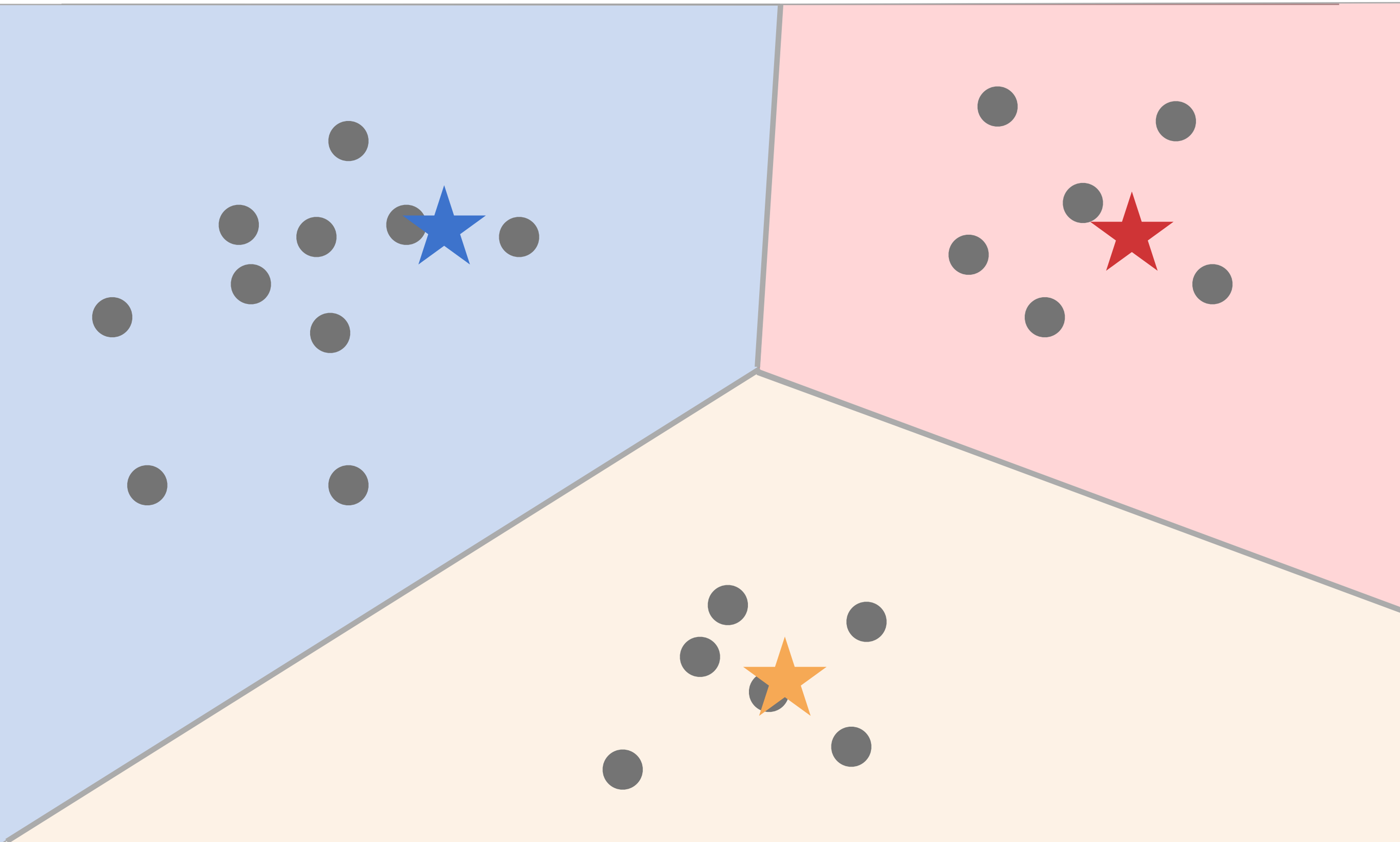
Now which data points are closest to each center?



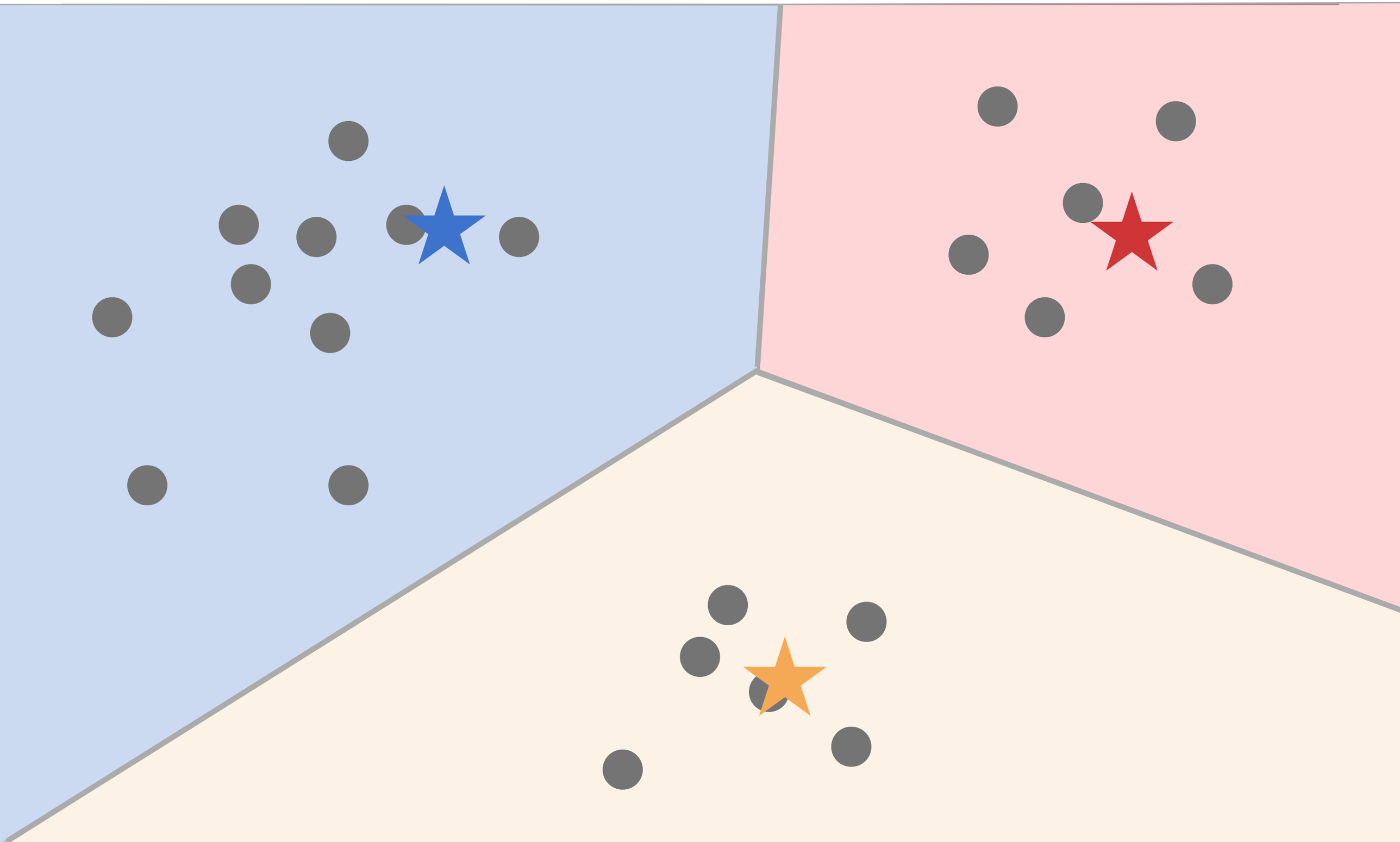
Now which data points are closest to each center?



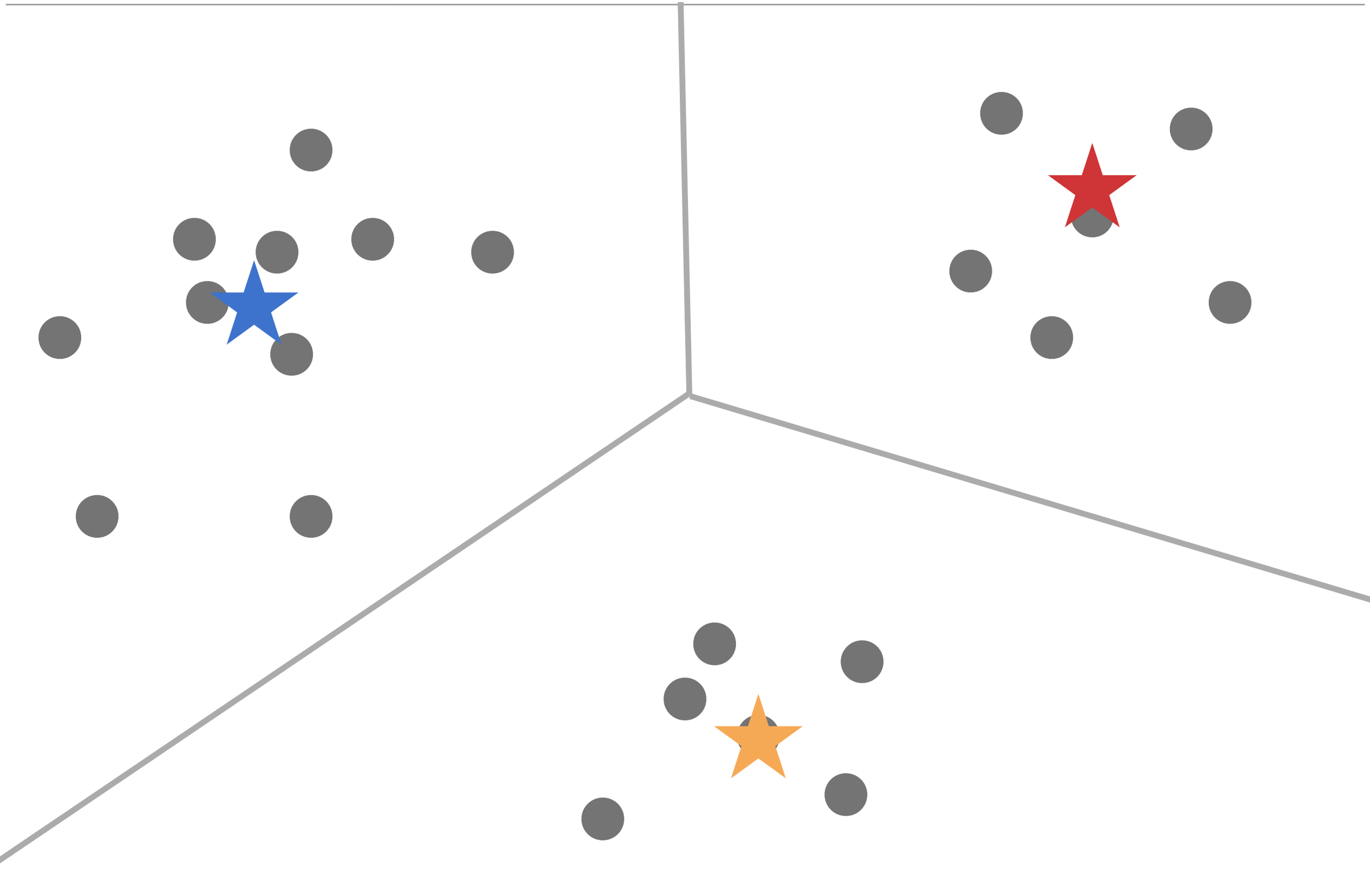
Redefine the clusters based on which center they're nearest



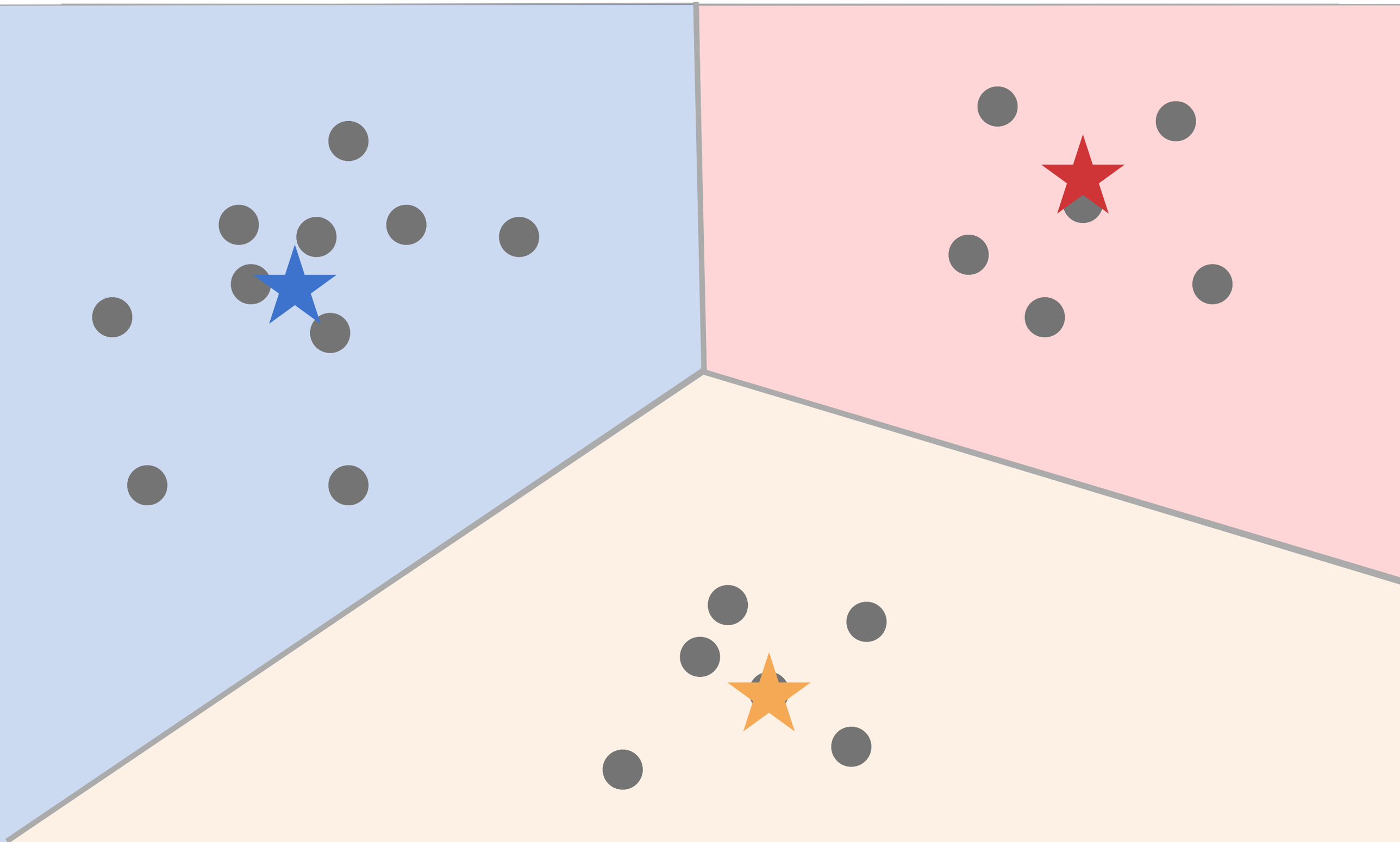
And repeat! Keep calculating the centers and redefining the clusters until they stop changing.



And repeat! Keep calculating the centers and redefining the clusters until they stop changing.



The results once the clusters and centers are fixed are your final k-means clusters.



K-means clustering

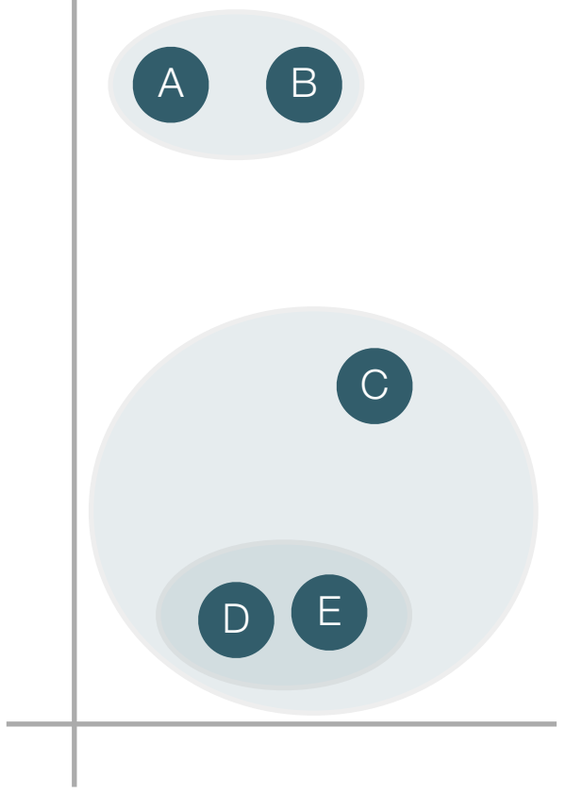
- Relatively efficient
- Can converge to local optima (e.g. depending on starting points)
- Have to specify k (number of clusters)
- Cannot make clusters with non-convex shapes
- Tends toward equal sized clusters
- How to handle categorical data? (e.g. can use k-modes)

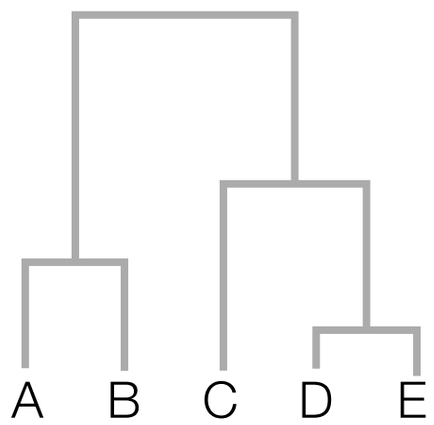
By Chire - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=11765684

Hierarchical clustering methods

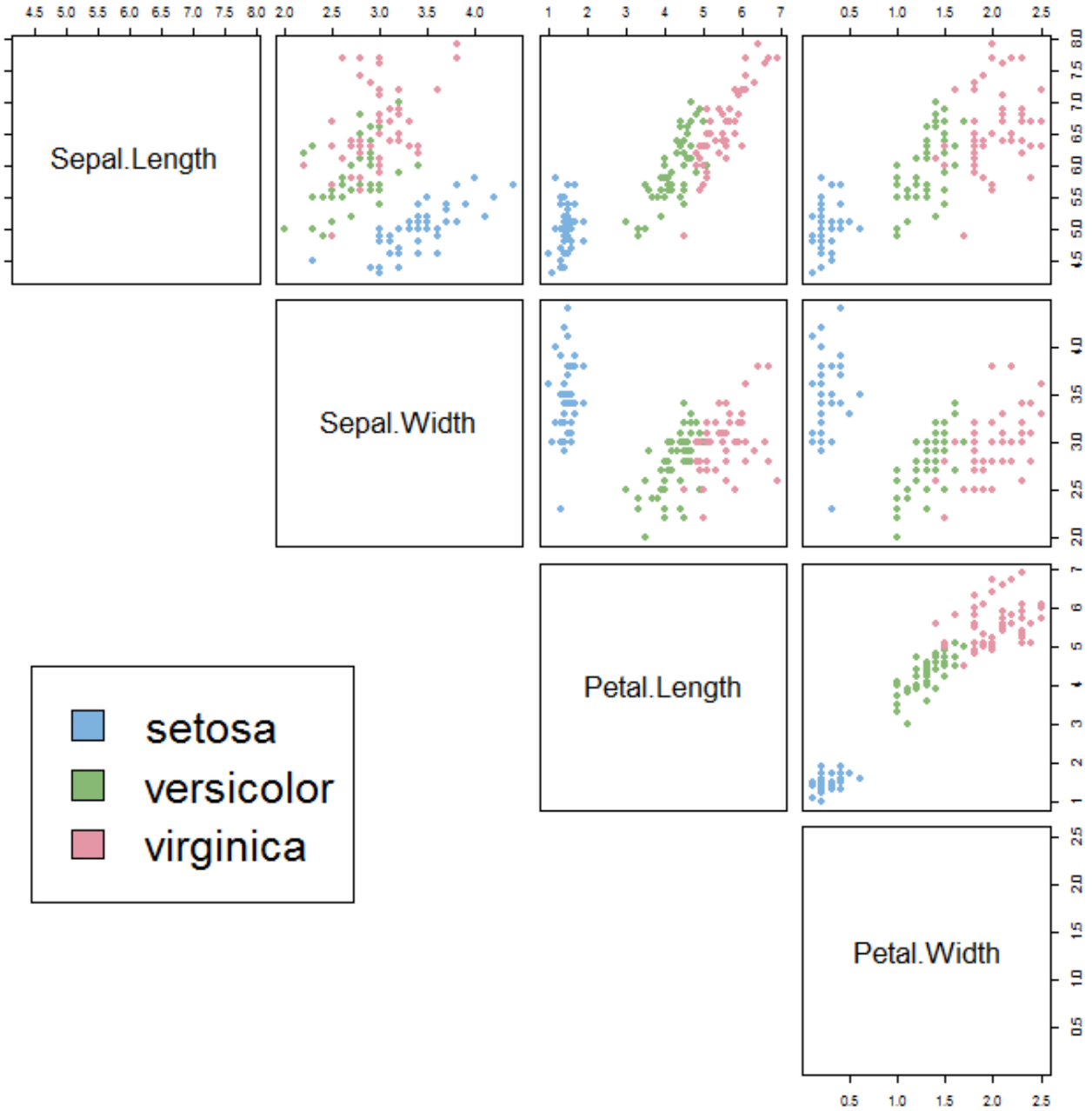
- Agglomerative approach to clustering
    - Starts with small clusters (e.g. individual points) and then merges based on distance
- Divisive approach does the reverse (all one cluster then split into smaller ones)
- Many different approaches with different distance measurements, etc.

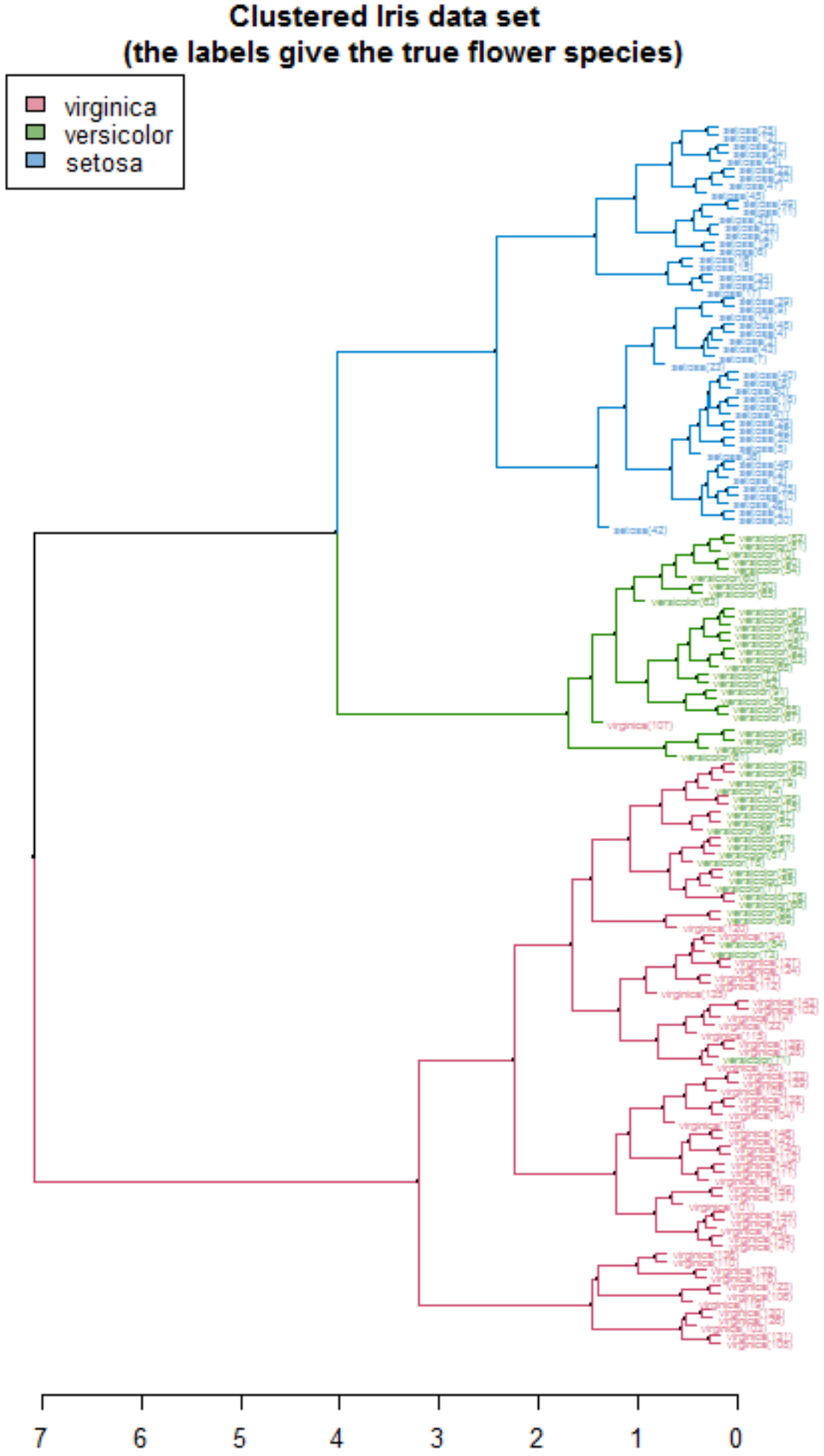
Hierarchical clustering example





- Start with all single point clusters
- Merge the two nearest clusters—forms a new cluster
- Merge the next two nearest clusters, etc.
- How to decide cluster distances? (What metric, do we use nearest point distance, furthest, centroid?)
- Capture clusters as a dendrogram—can choose resolution of clusters as desired





https://cran.r-project.org/web/packages/dendextend/vignettes/Cluster_Analysis.html

Hierarchical clustering

- Slow for larger data sets
- Useful for finding substructures/subclusters in data
- Assumes every data point is relevant/part of the clusters
- How to choose level of granularity?

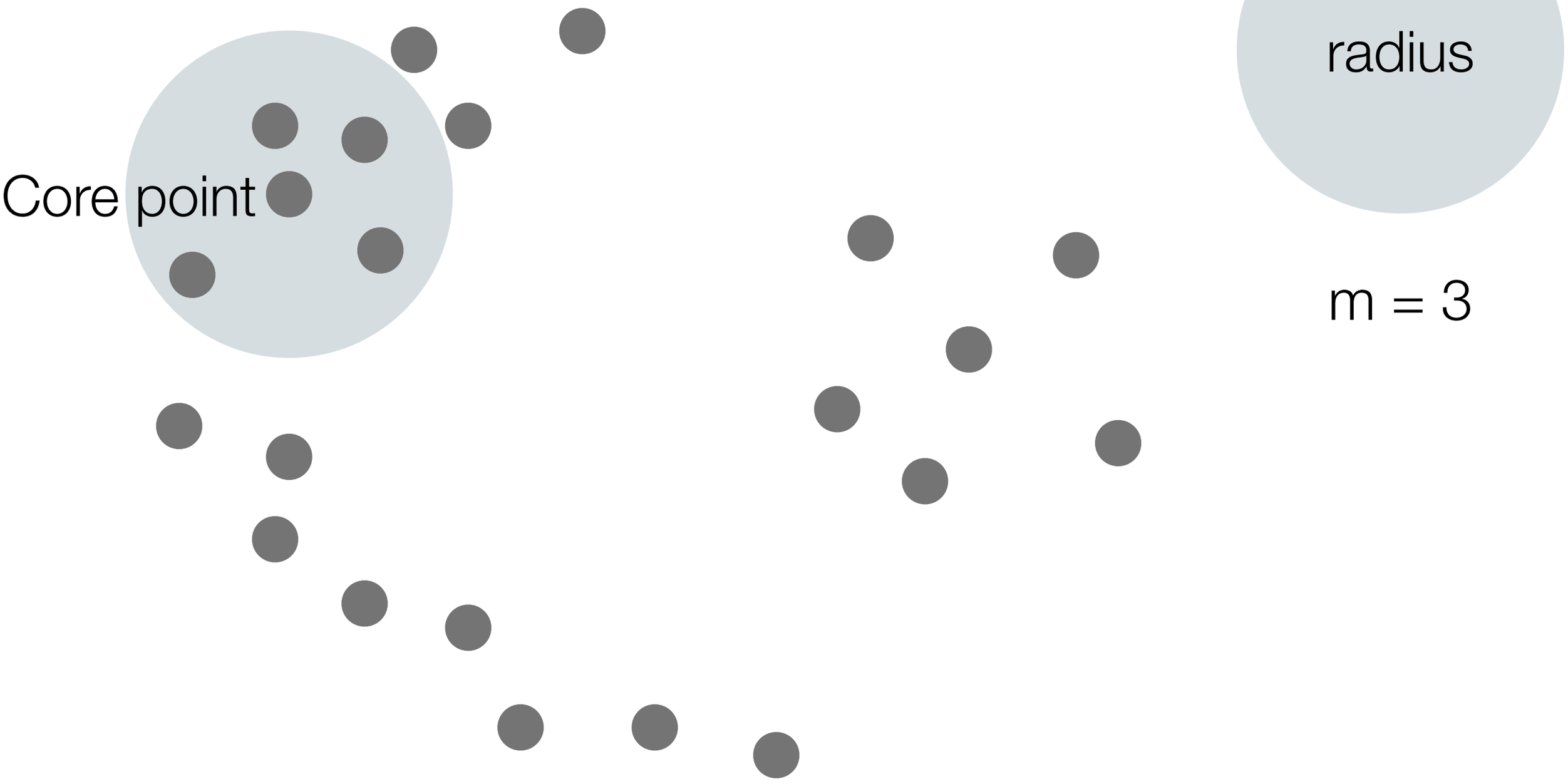
Density-based clustering

- Decides clusters based on density of points
- Not every point need be assigned a cluster—some can be considered noise or outliers
- One of the most commonly used algorithms is DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

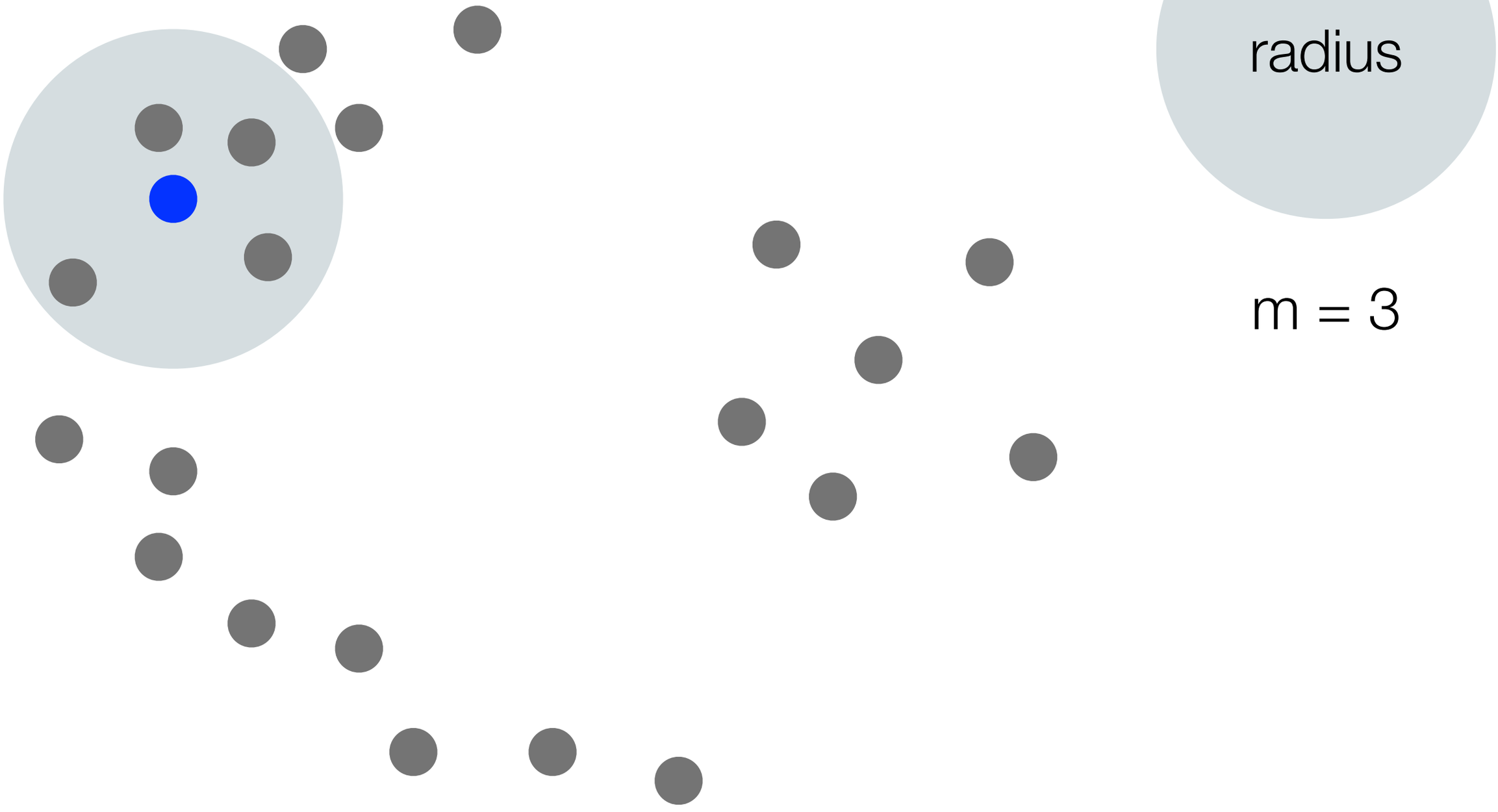
DBSCAN

- Choose a radius *r* and a minimum number of points *m*
- Classify each point as a:
  - **Core point** - has at least *m* other points within radius *r*
  - **Border point** - does not have *m* points within radius *r*, but is **reachable** a core point *p* - i.e. can be connected to data point *p* by a chain of core points each within radius *r* of the next point
  - **Outlier** - neither core nor border

DBSCAN example



DBSCAN example

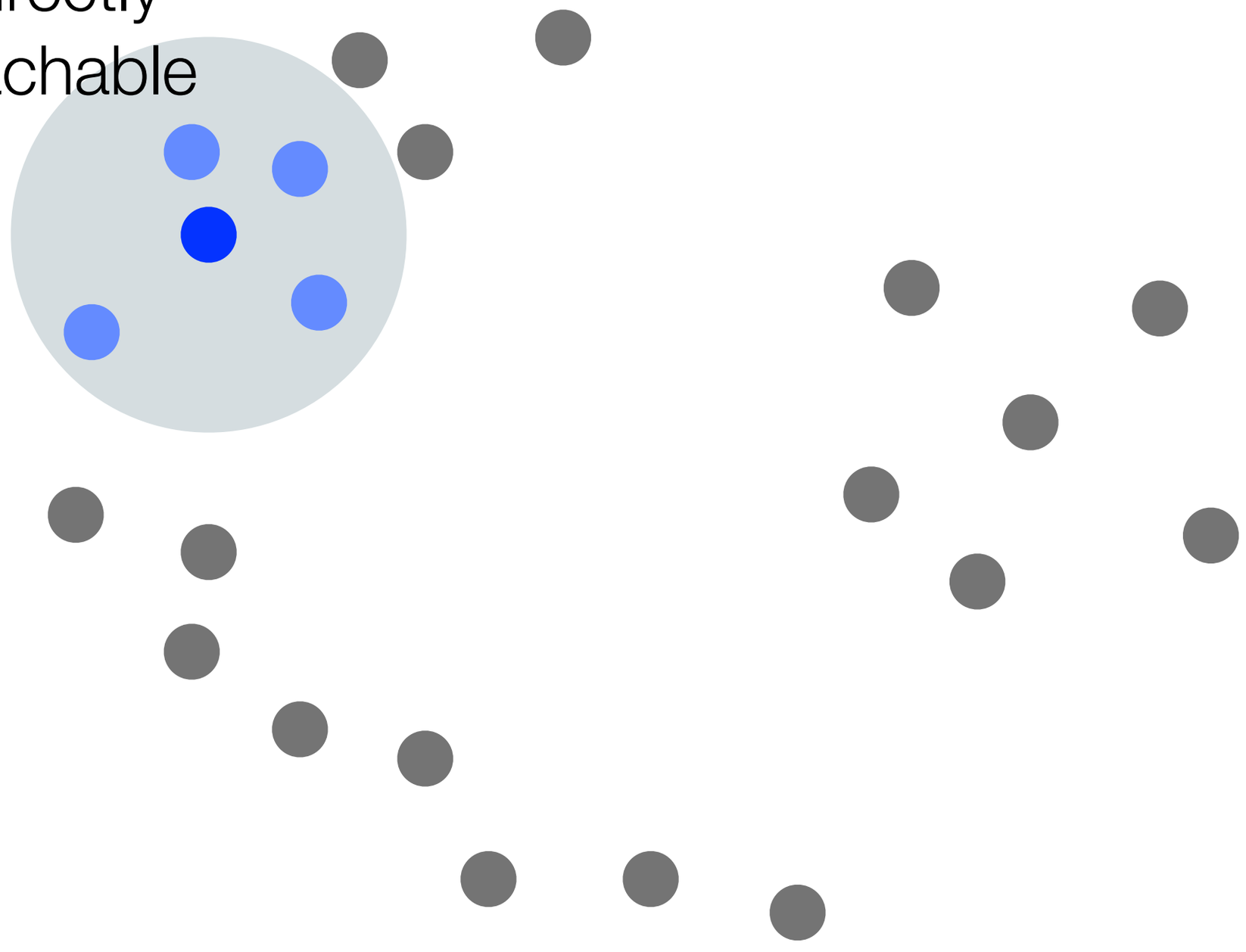


DBSCAN example

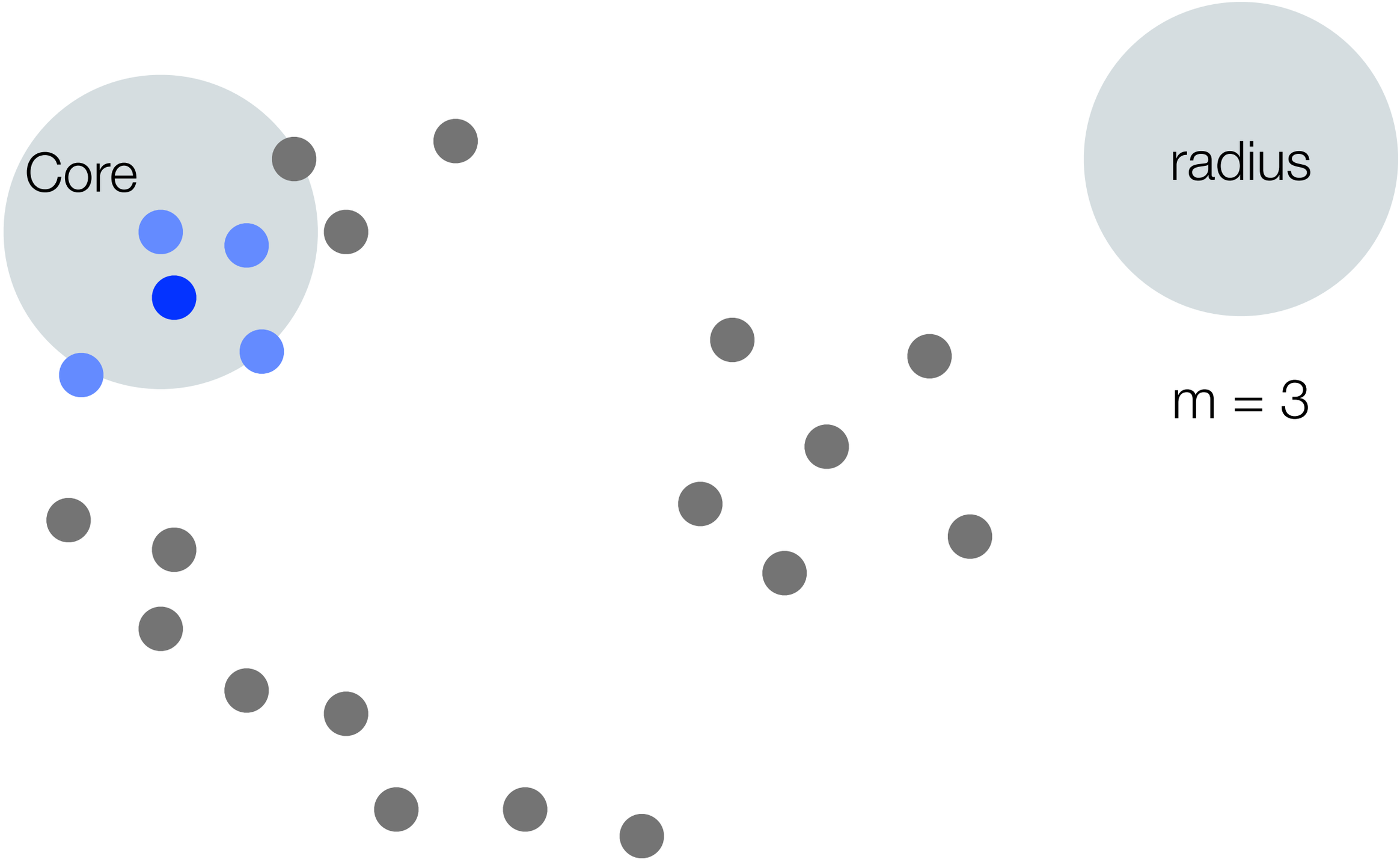
Directly reachable

radius

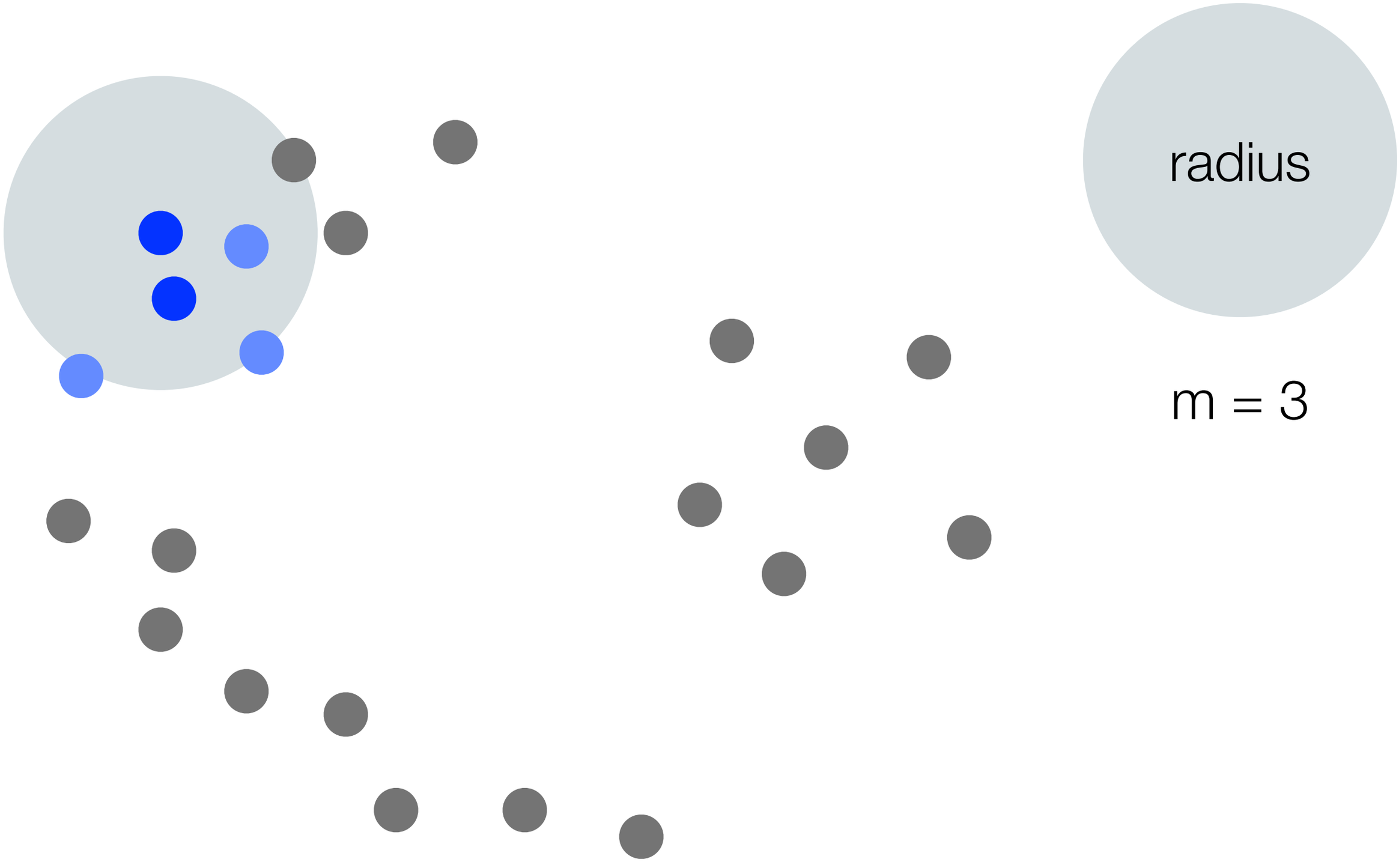
m = 3



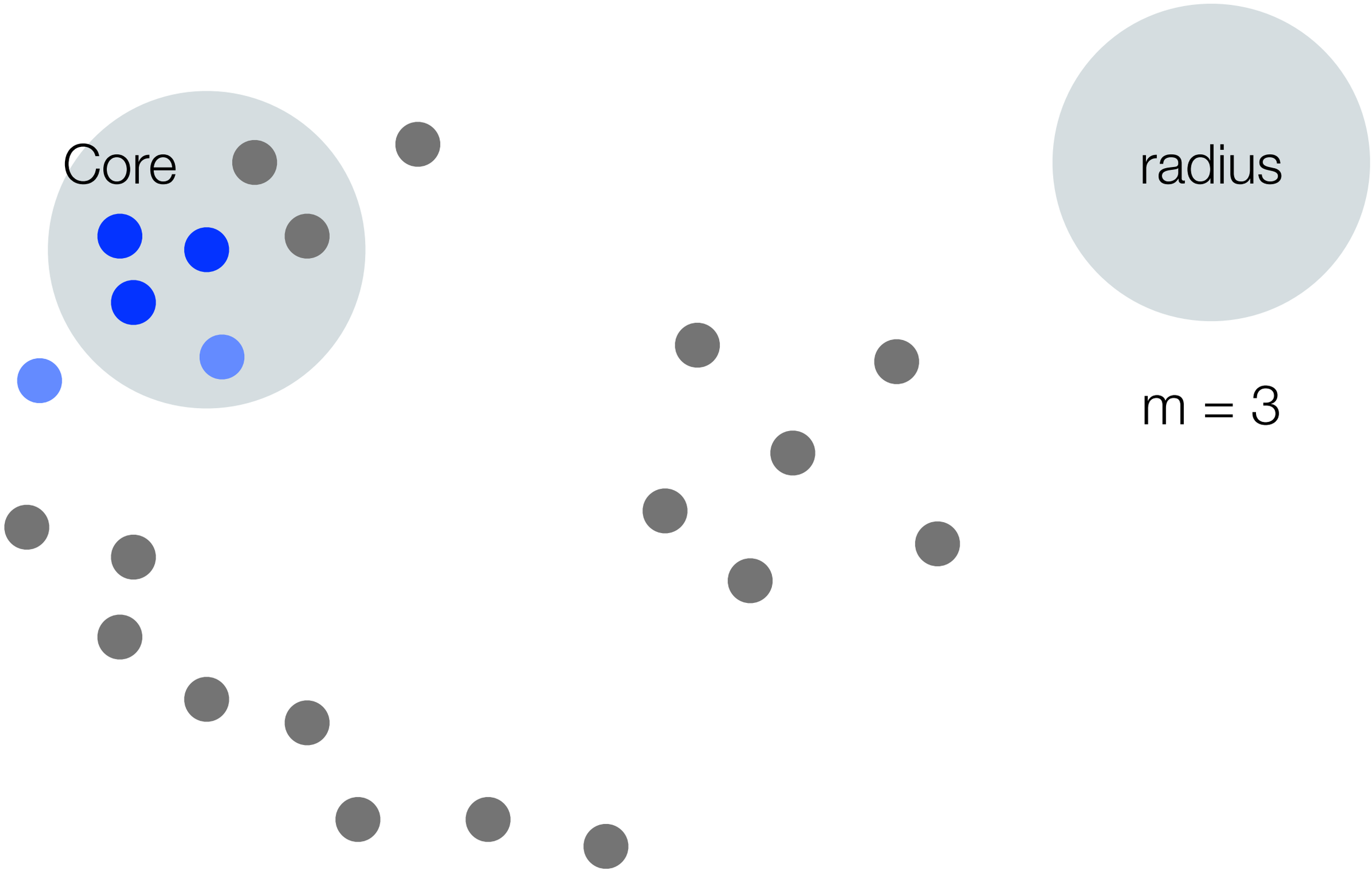
DBSCAN example



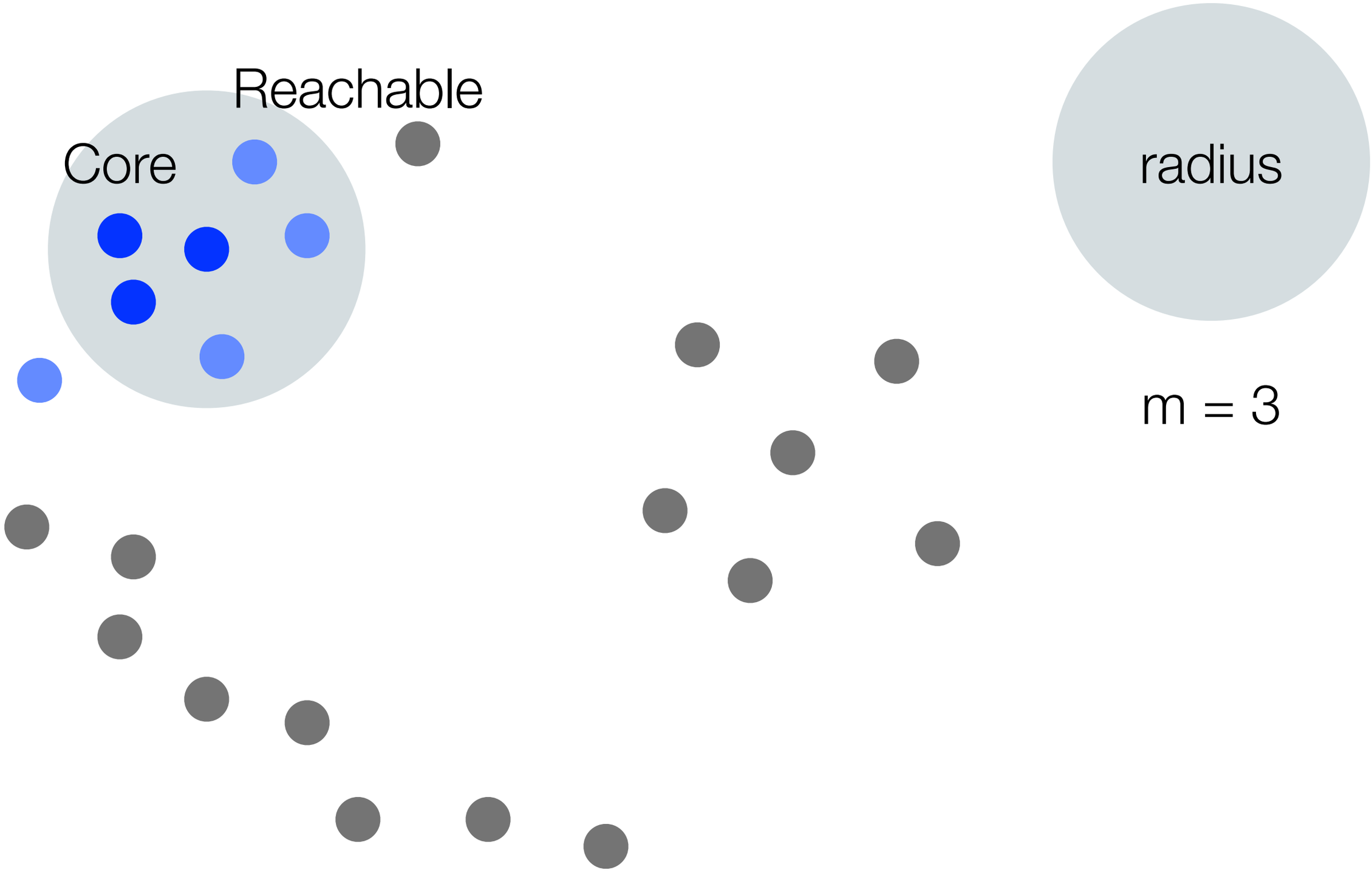
DBSCAN example



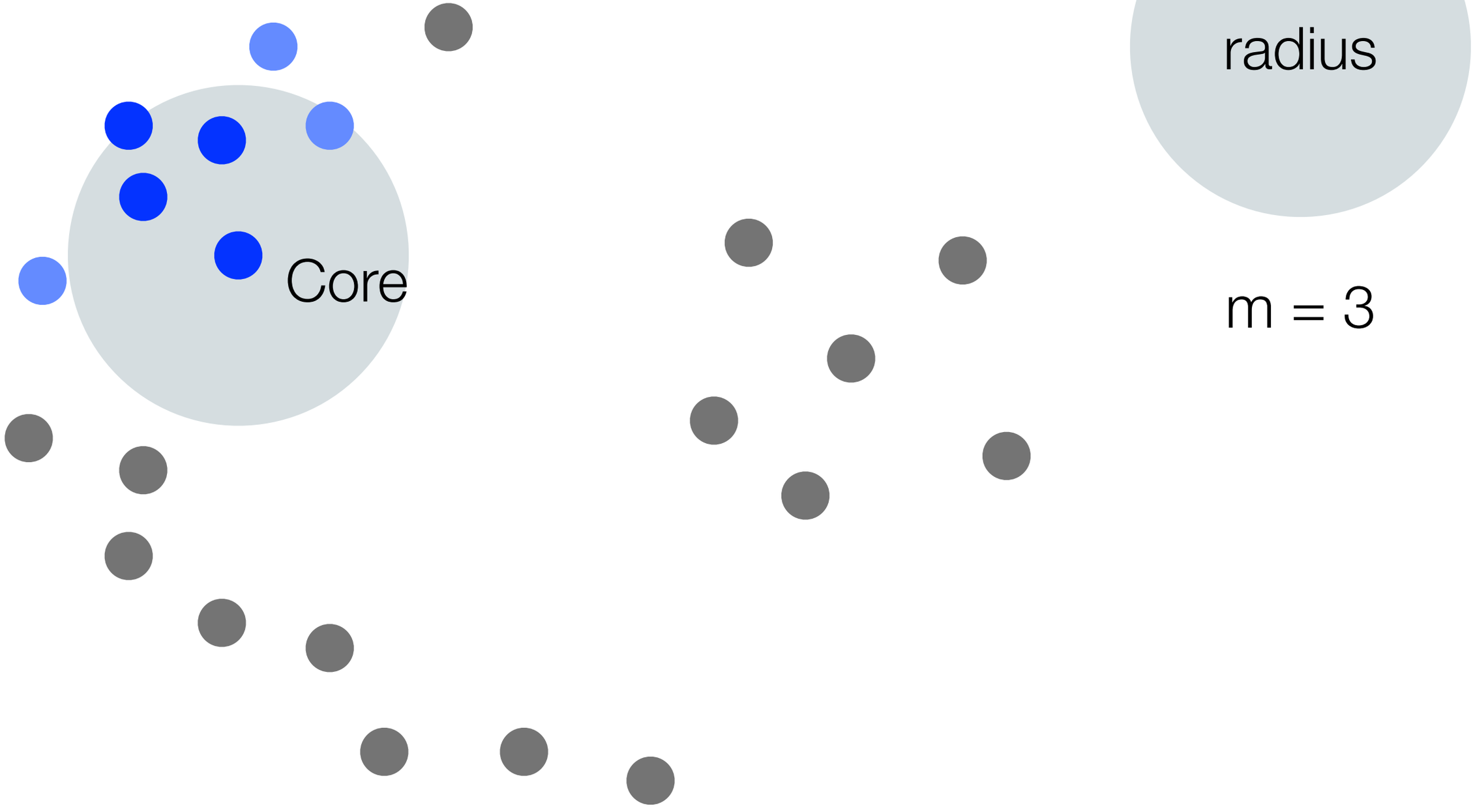
DBSCAN example



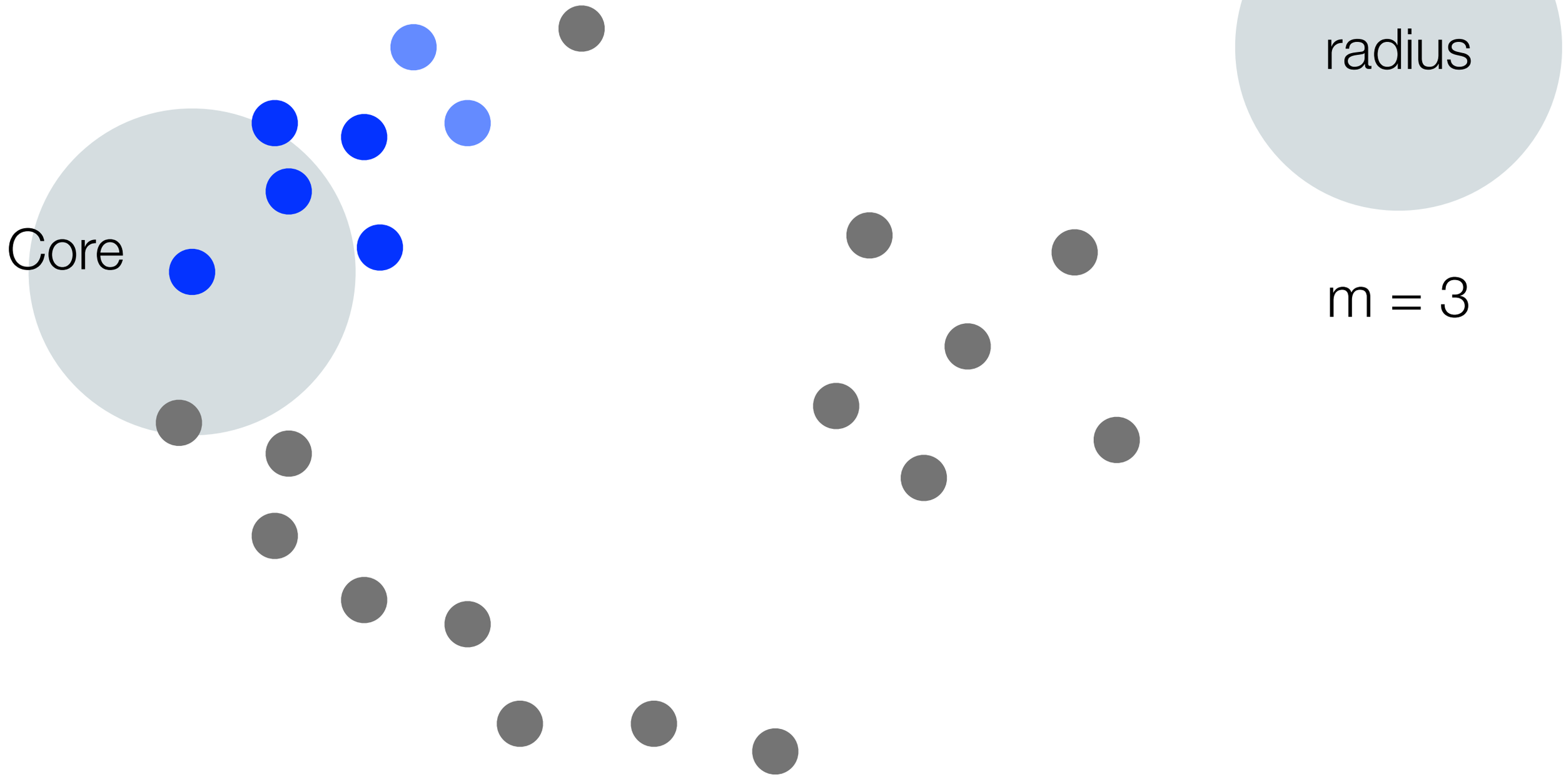
DBSCAN example



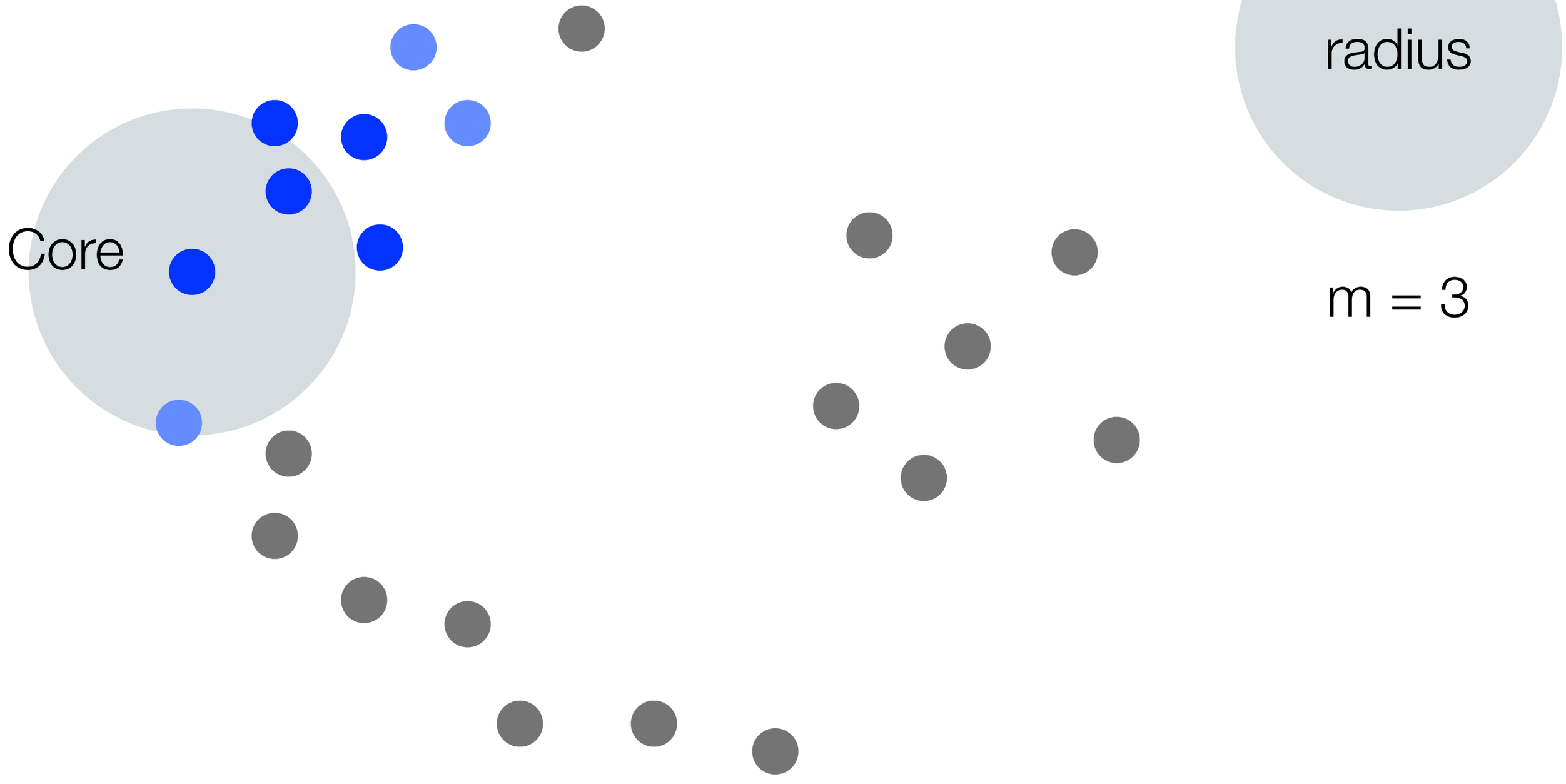
DBSCAN example



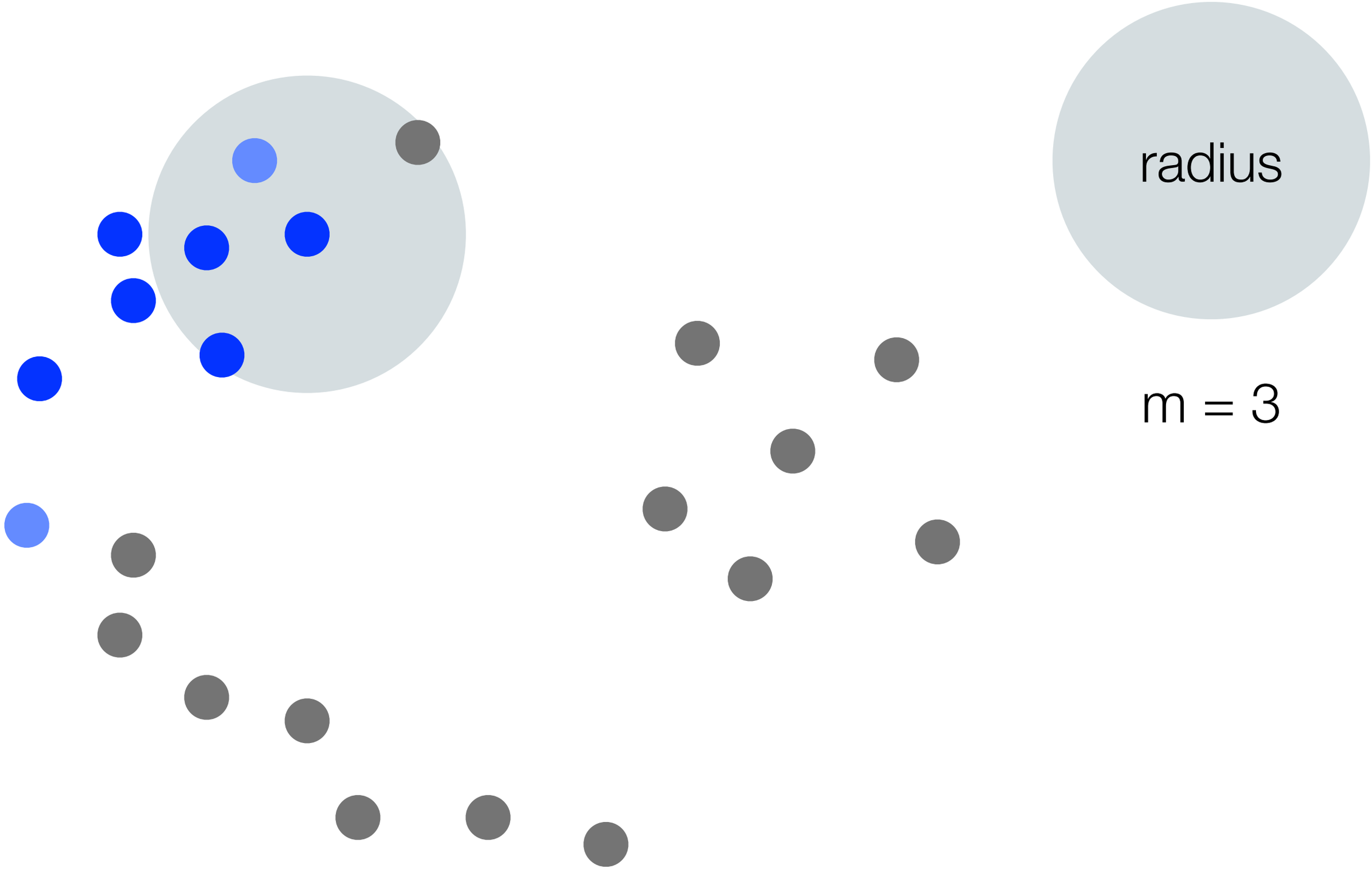
DBSCAN example



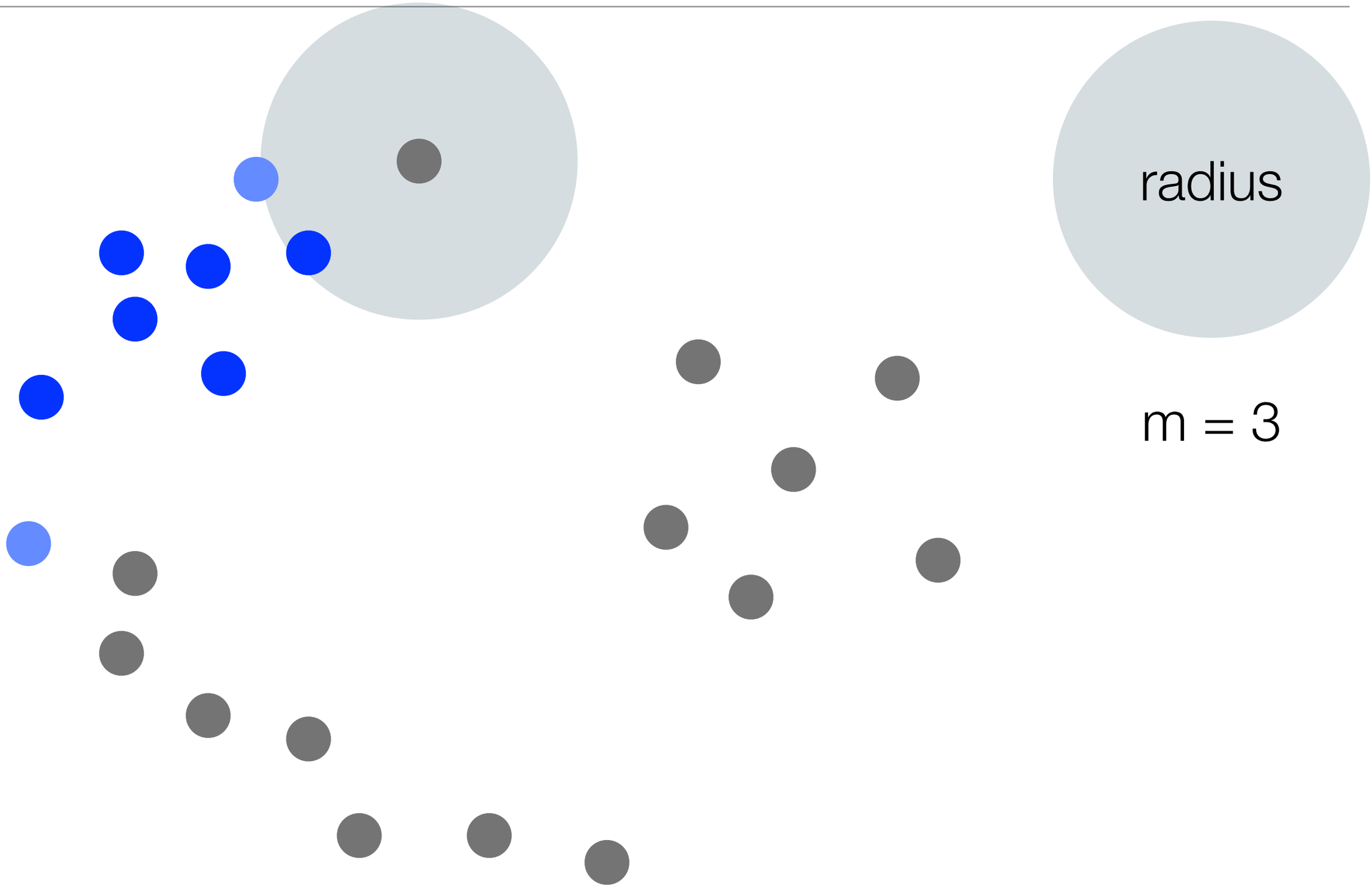
DBSCAN example



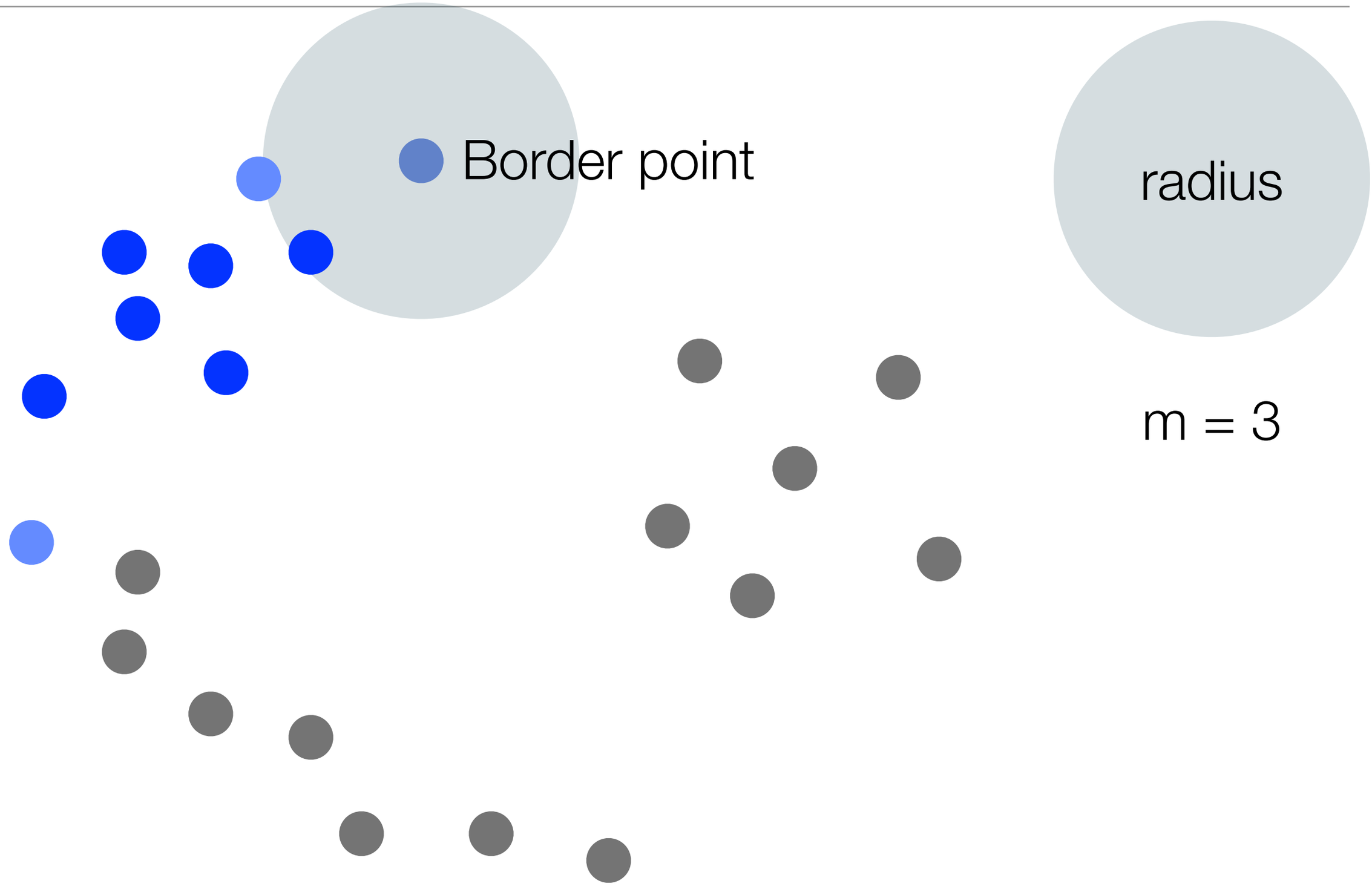
DBSCAN example



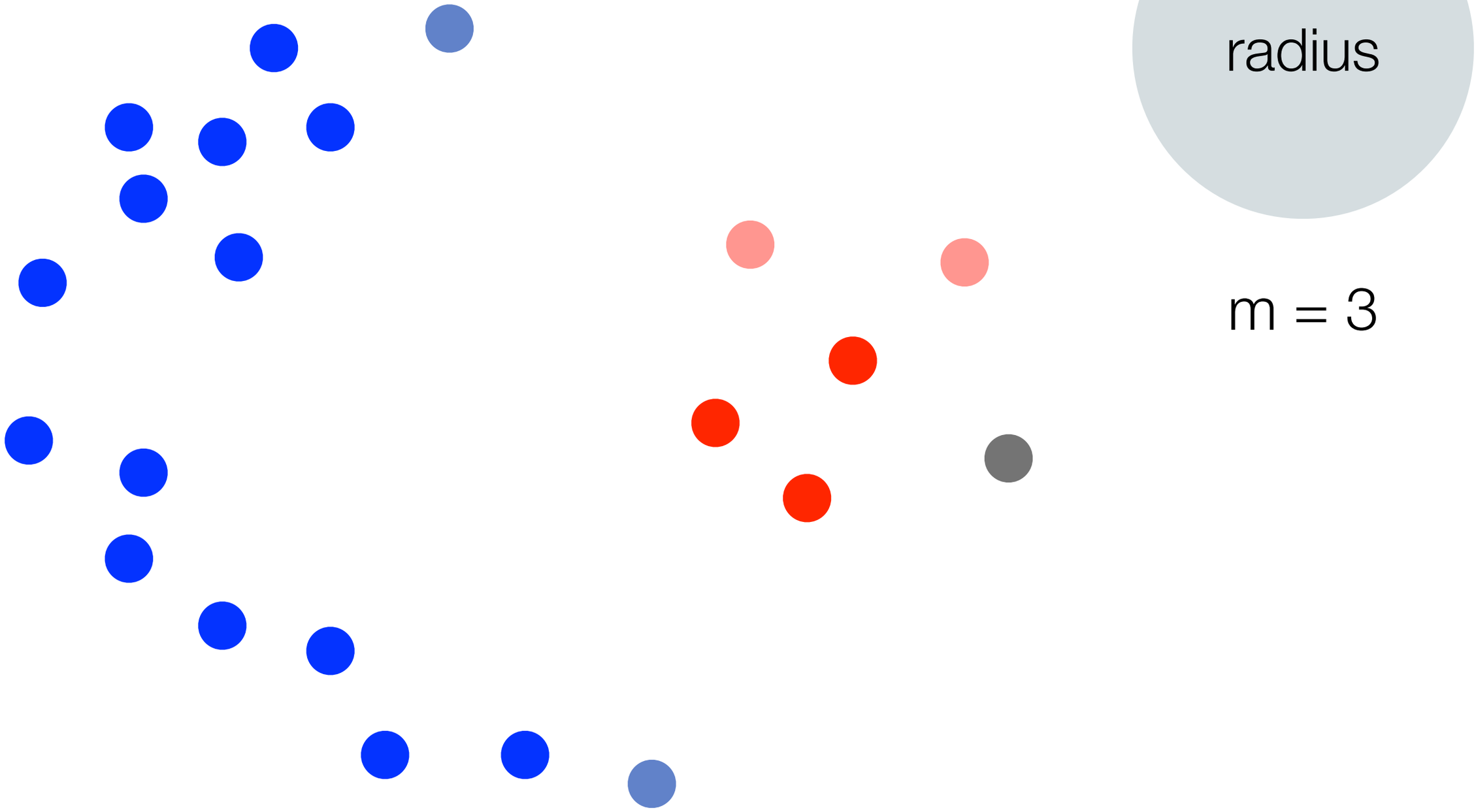
DBSCAN example



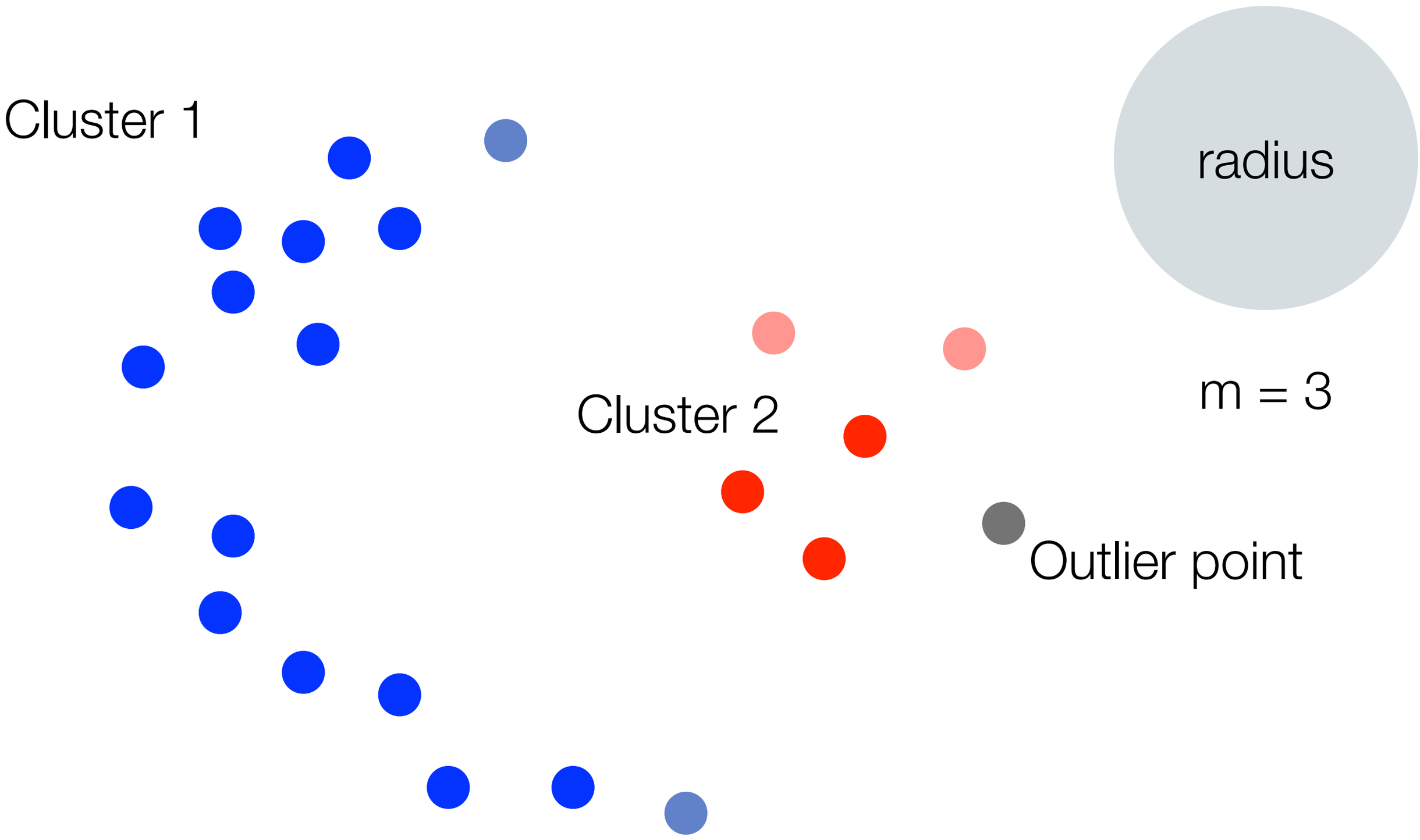
DBSCAN example



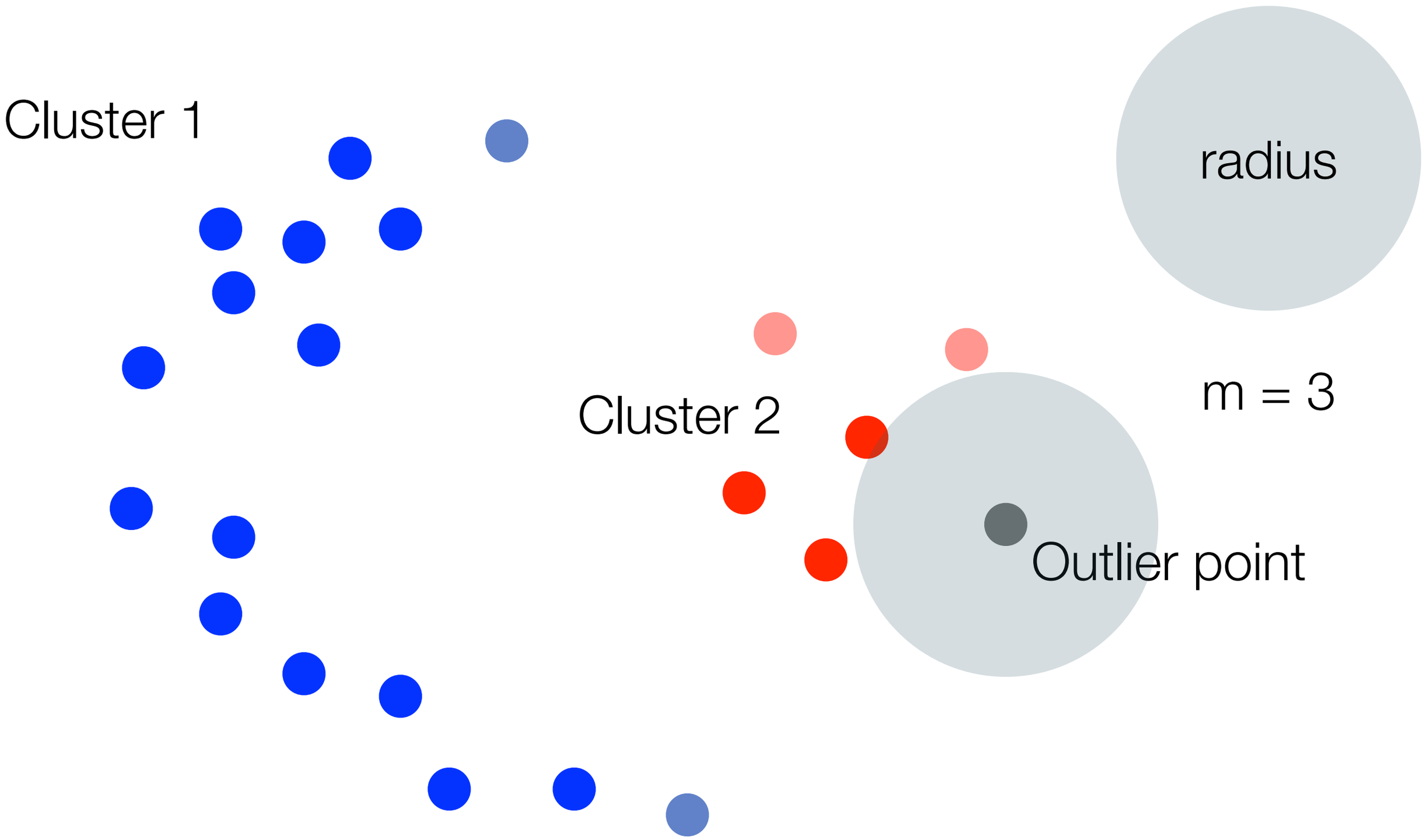
DBSCAN example



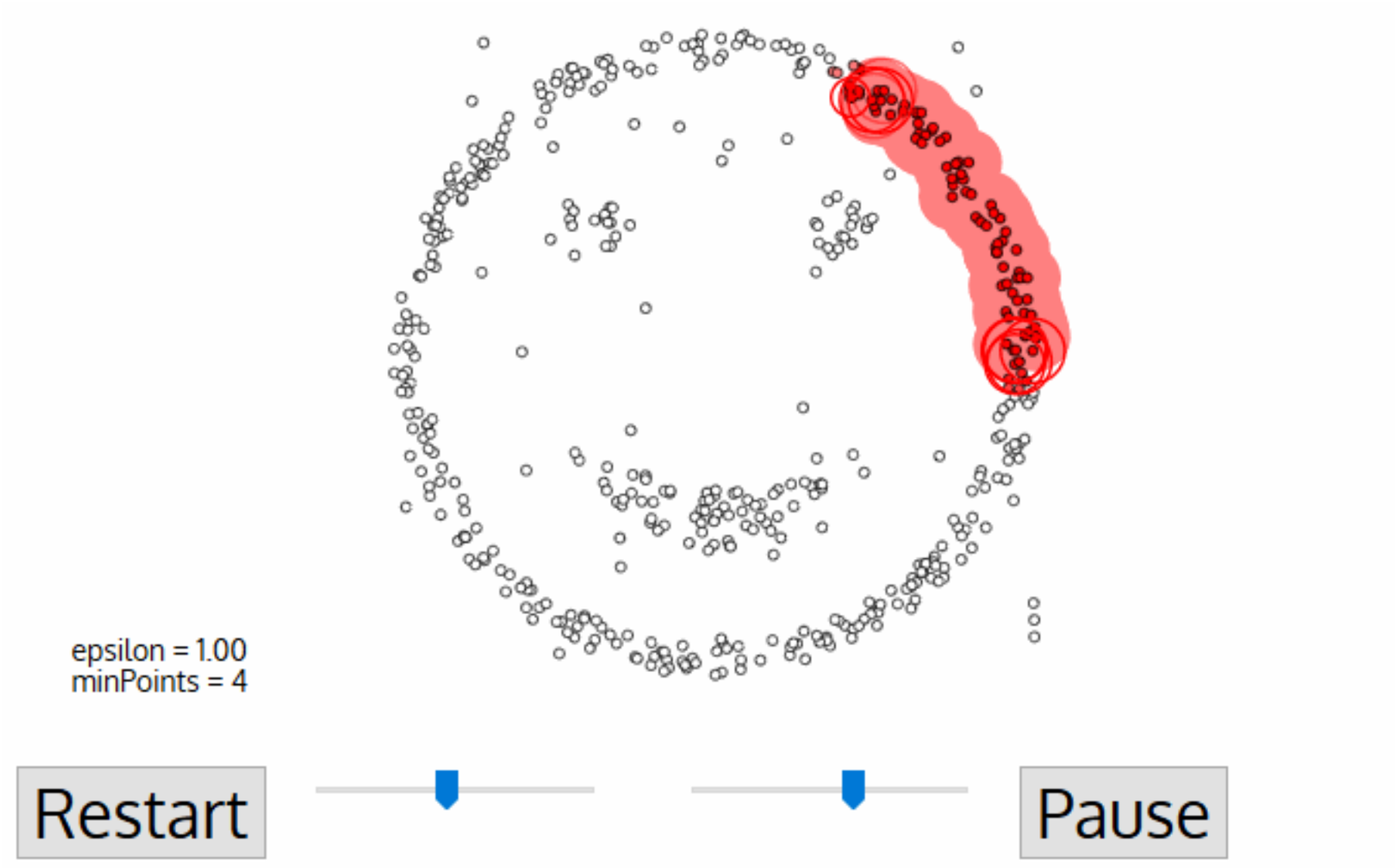
DBSCAN example



DBSCAN example



DBSCAN



https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68

DBSCAN

- Can find non-convex clusters
- Automatically determines number of clusters needed
- Not every point goes into a cluster (handles outliers/noise; however can be a drawback if you want to assign all points to a cluster)
- Tends to find/work best with clusters of similar density
- How to choose radius & min points? There are rules of thumb but can be tricky! Often use min points = 2 x dim, for radius, can us elbow plot of a k-distance graph, but harder to say)

Model based methods: Gaussian Mixture Models

- Assumes the data points come from a combination of multivariate gaussians

- This seems restrictive but is often no more so than other methods (e.g. k-means in some sense assumes a centroid and resulting Voronoi diagram govern the data)

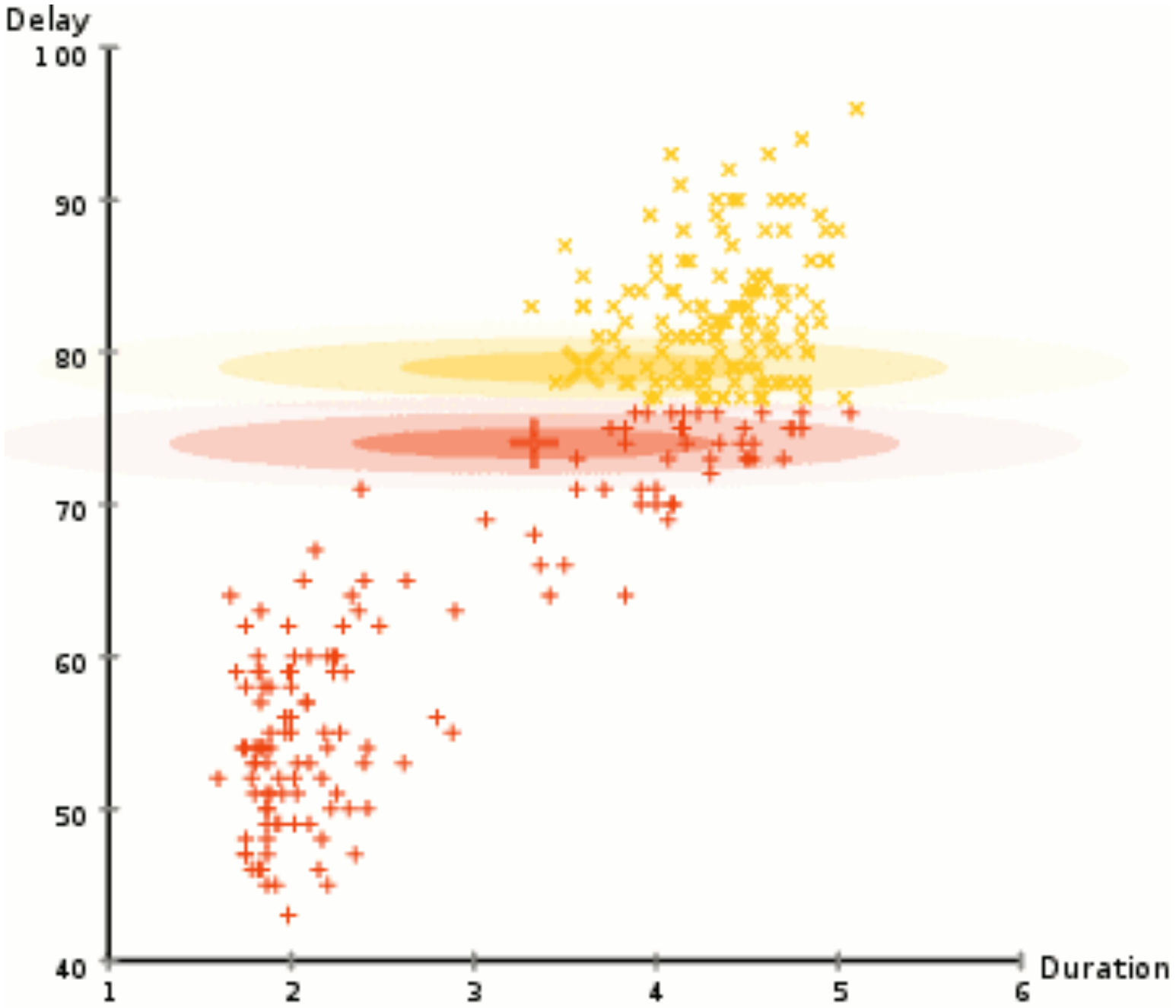
- Each data point has a probability of belonging to each cluster

- Often fit via expectation maximization (a type of maximum likelihood approach)

Model based methods: Gaussian Mixture Models

- Select number of clusters (number of gaussians to fit)
- Randomly initialize them (or better yet, use a method to pick a good starting guess)
- Compute the probability that each data point is in each cluster (based on the value of the gaussian at that point)
- Compute new parameters (μ,σ) for each gaussian that maximize this probability
- Repeat last two steps above until convergence

Model based methods: Gaussian Mixture Models



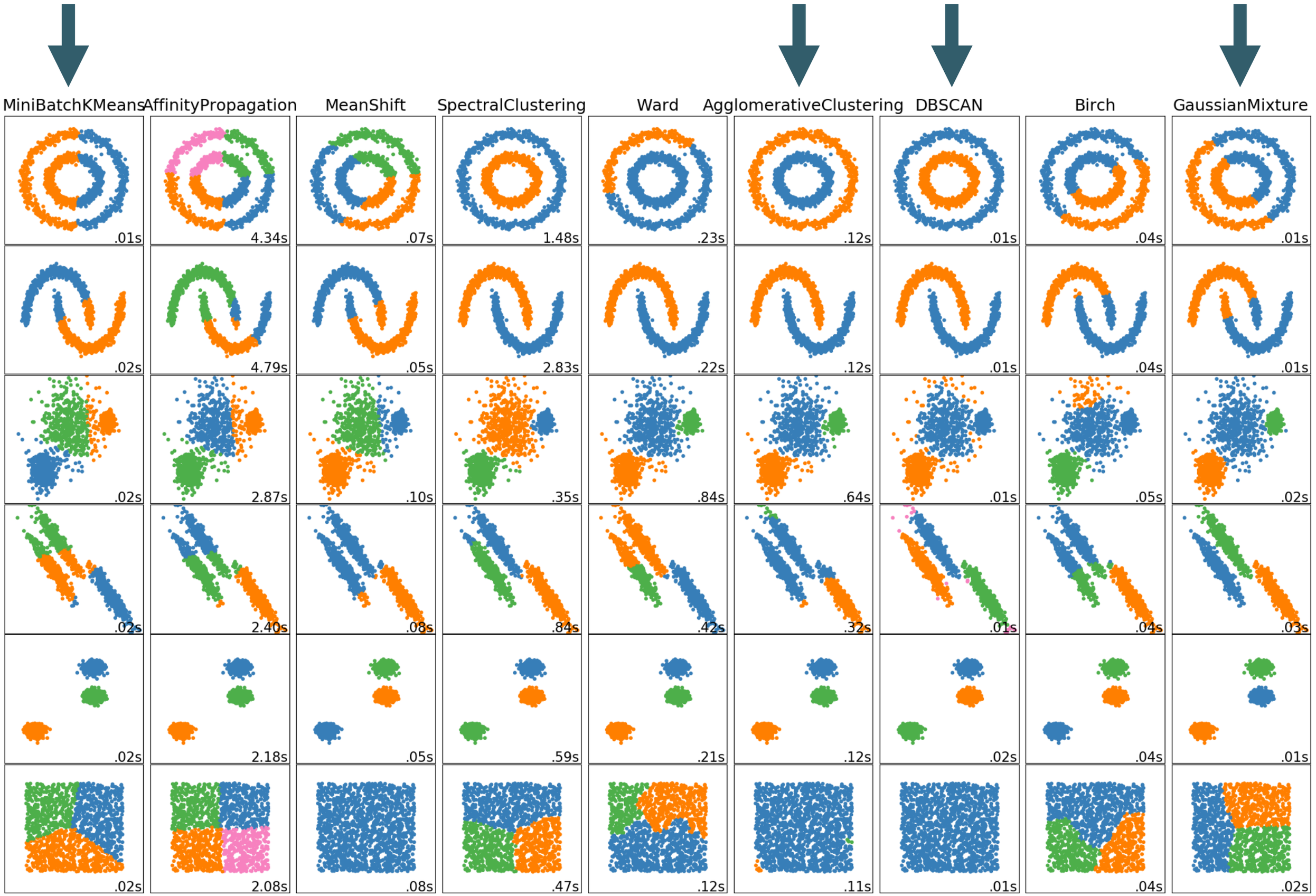
https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68

Clustering methods

- Many different approaches! These are just a few examples

- Different methods behave better/worse on different data sets

- Testing how well a clustering method behaves can be difficult, especially in high dimensions and/or without ground truth information



https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68

Resources

- https://en.wikipedia.org/wiki/Cluster_analysis
- https://en.wikipedia.org/wiki/DBSCAN
- https://medium.com/predict/three-popular-clustering-methods-and-when-to-use-each-4227c80ba2b6
- https://blog.dominodatalab.com/topology-and-density-based-clustering/
- https://shapeofdata.wordpress.com/2014/03/04/k-modes/
- https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68

Supervised learning example: **classification**

Supervised learning

$$\vec{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix}$$

Input data vector
vector of features/attributes/predictors/ covariates

$(\tilde{x}_1, \tilde{y}_1), \ldots, (\tilde{x}_n, \tilde{y}_n)$

Training data
input-output pairs - i.e. data vectors ($\tilde{x}_i$) with labels ($\tilde{y}_i$)

Goal: given new input data $\vec{x}$, predict the output $\vec{y}$

Classification

- A form of supervised learning
- Starts with a known set of classes
- Given training data that is labeled with which class it belongs to, predict which class new (unlabeled) data belongs to

Puppy or Bagel?

Labradoodle or Fried Chicken?

Chihuahua or Muffin?

Shar-Pei or Towel?

Shiba Inu or Marshmallow?

Parrot or Guacamole?

Classification

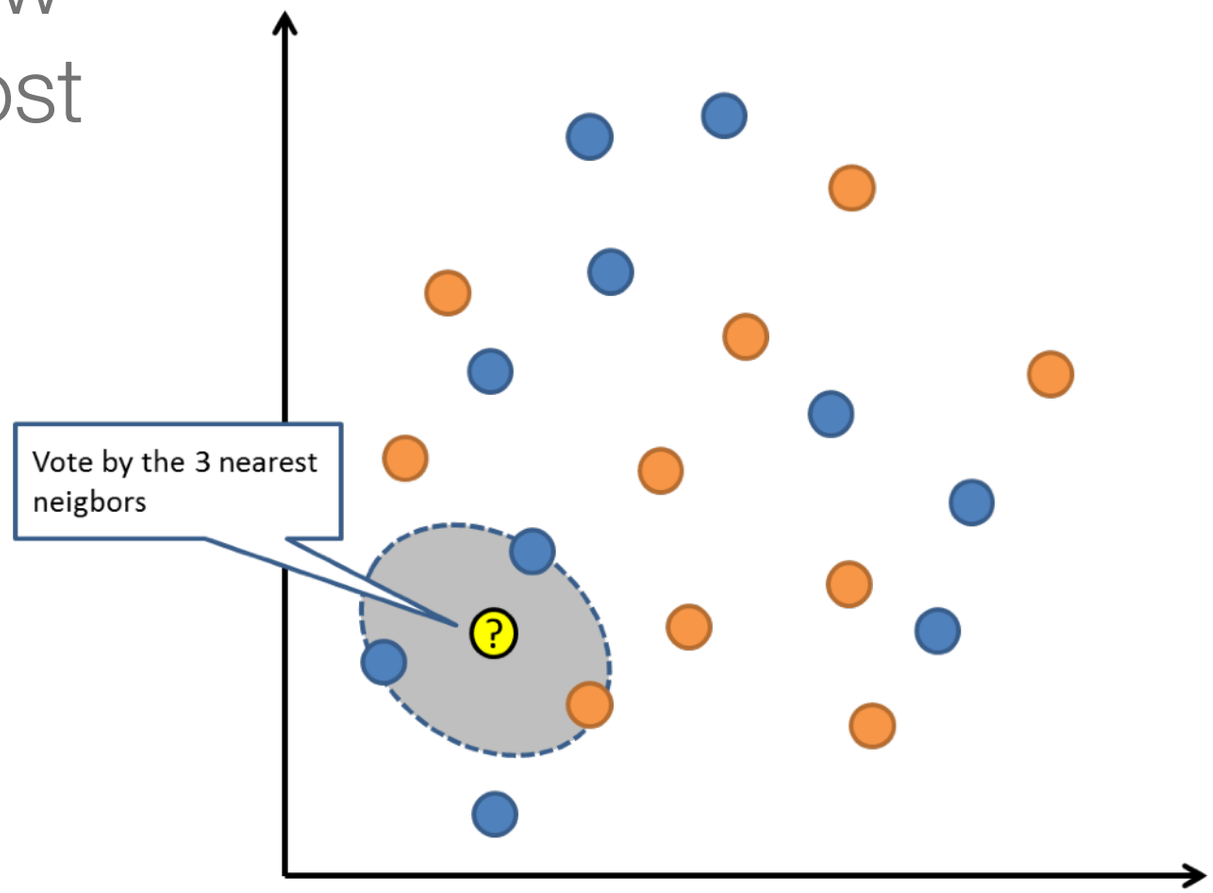
- Many (many!) different kinds
- Based on a distribution model or not
- Linear or nonlinear
- ‘Lazy’ vs ‘eager’ learners
  - Lazy learners (related: instance-based learning) load all the training data and then use the training data when new inputs come in to decide (often fast to train, but slow to operate on new data)
  - Eager learners build the classifier on the training data first, then just apply the classifier (e.g. some model) on new input data (often slow to train, but fast on new data)

Classification

- K-nearest-neighbors
- Bayes classifier-based methods
  - Naive Bayes
  - Linear discriminant analysis
  - Logistic regression
- Support vector machines
- Decision trees - relatedly: random forests
- Neural networks

k-Nearest-Neighbors (kNN)

- Decide output (class) for new data point based on the most common class among the nearest k neighbors in the training set



- Distance can be Euclidean, Hamming distance, etc.

- Weighted kNN - weight the nearest neighbors by their distance (e.g. 1/d)

https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623

k-Nearest-Neighbors (kNN)

- No explicit training phase, but the k nearest neighbors can be viewed as the training data

- Lazy learner/instance-based learner

Bayes Classifiers

- How to think about classifiers probabilistically?
- Suppose we have inputs and outputs:

$$X \in \mathbb{R}^d \qquad\qquad Y \in \{1, \ldots, K\}$$

- The joint distribution P_{XY} can be broken down:

$$P_{XY} = P_{X|Y} P_Y$$

Conditional distribution

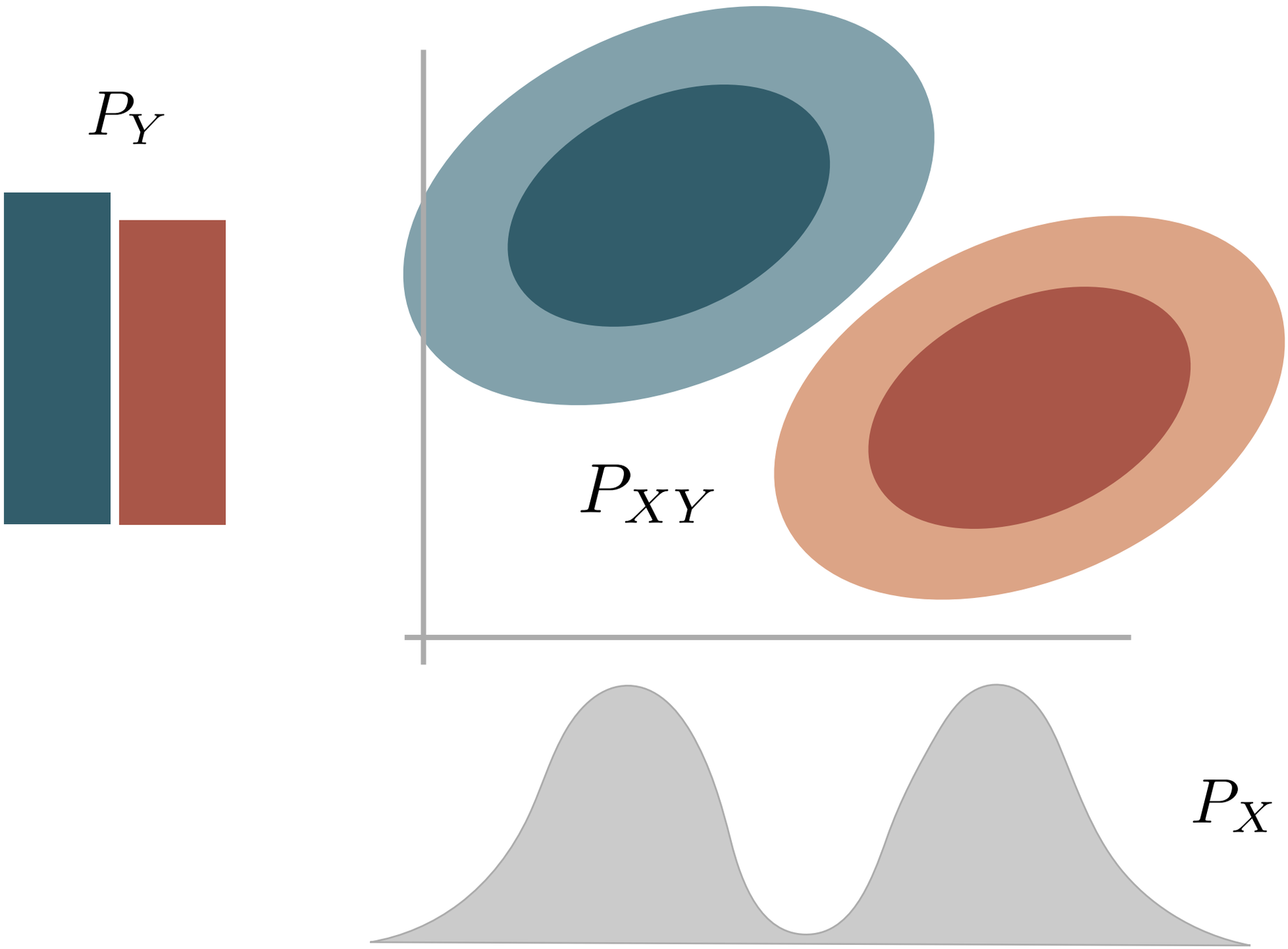
Marginal distribution of Y (prior class distribution)

$$P_{XY} = P_{Y|X} P_X$$

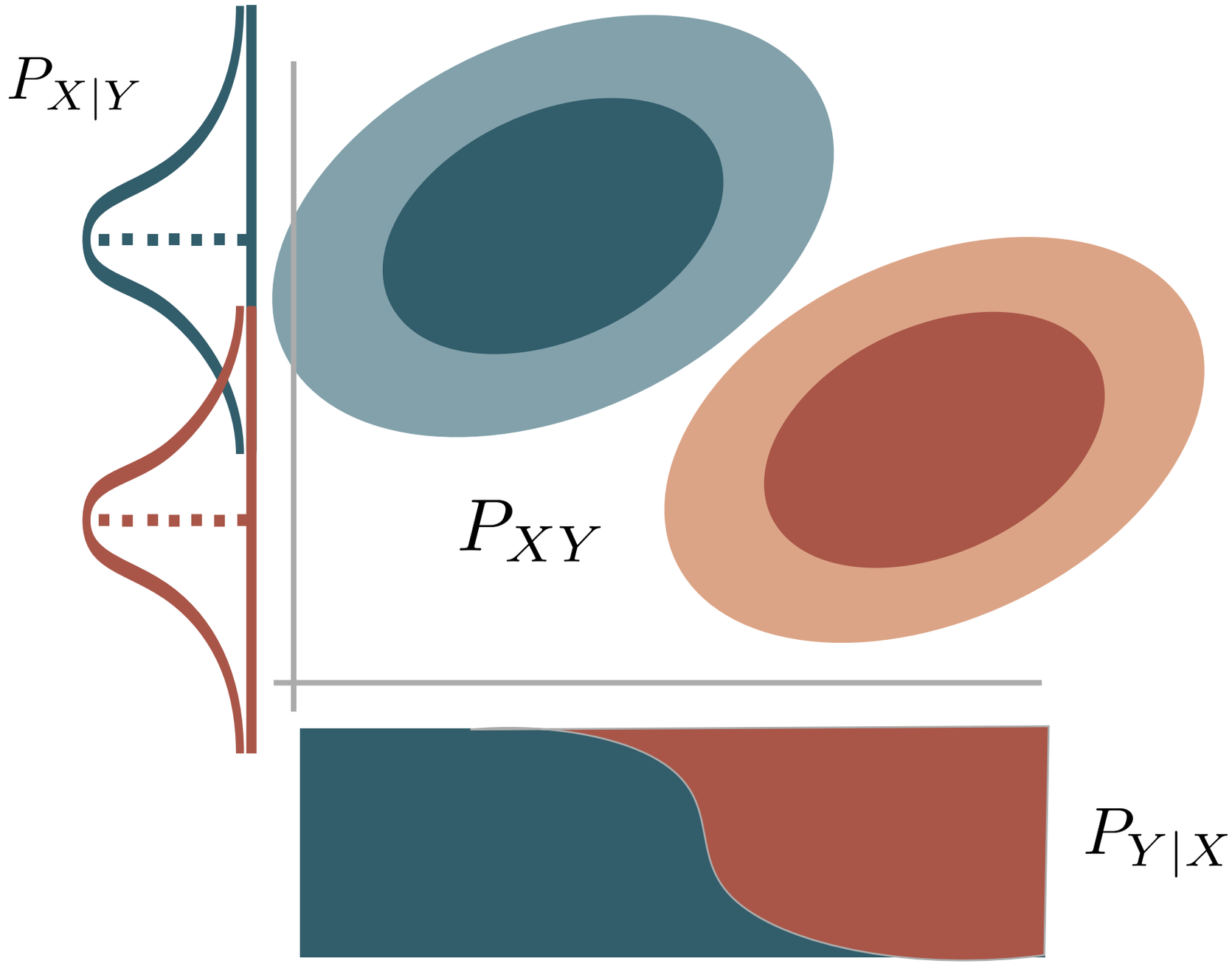
Posterior of Y given X

Marginal of X

Bayes classifiers



Bayes classifiers



Bayes classifiers

- What is a classifier? A function $f : \mathbb{R}^d \to \{1, \ldots, K\}$ that takes input data and returns an output (class)

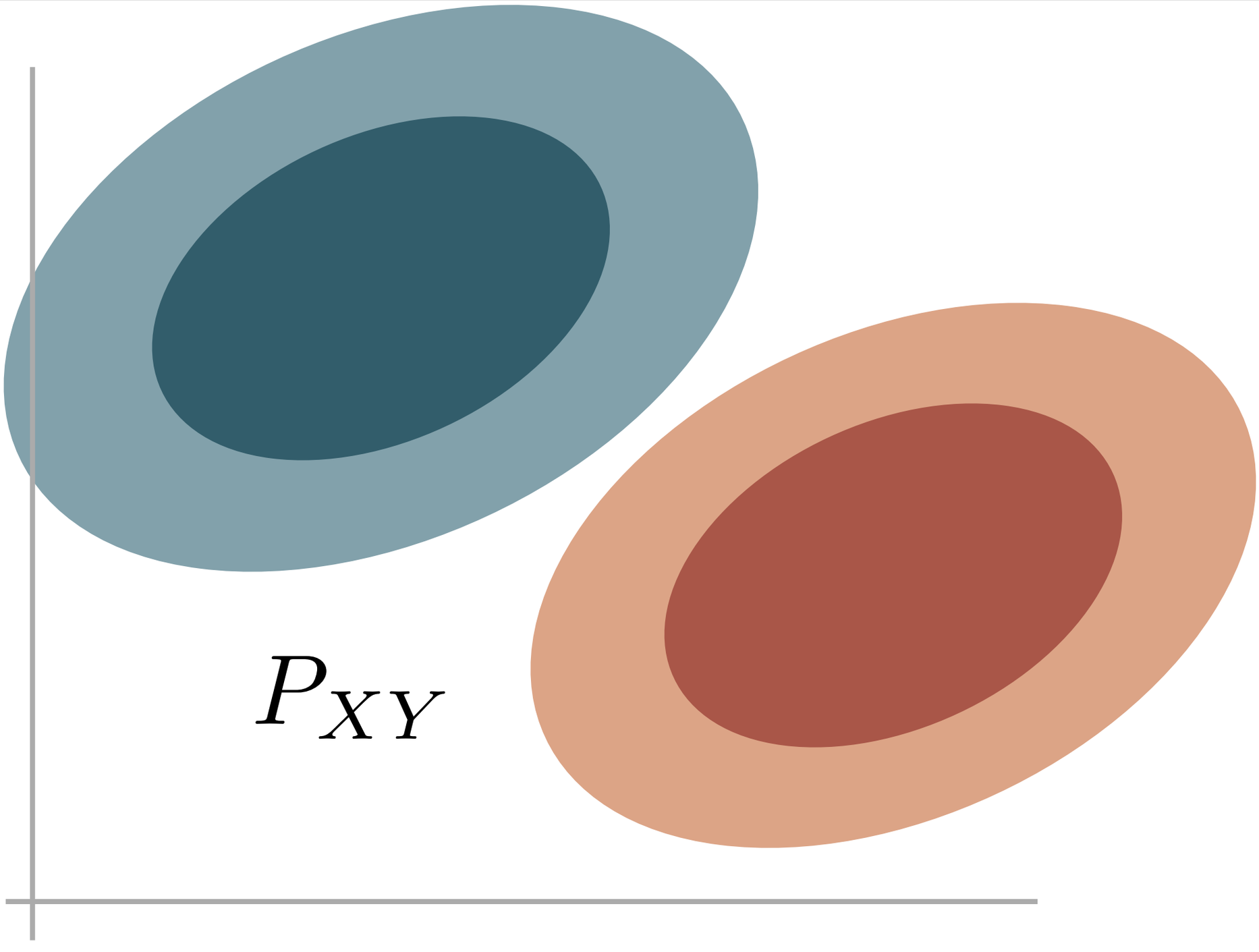
- How to choose the best classifier?

- One way is to minimize the classifier error (risk):

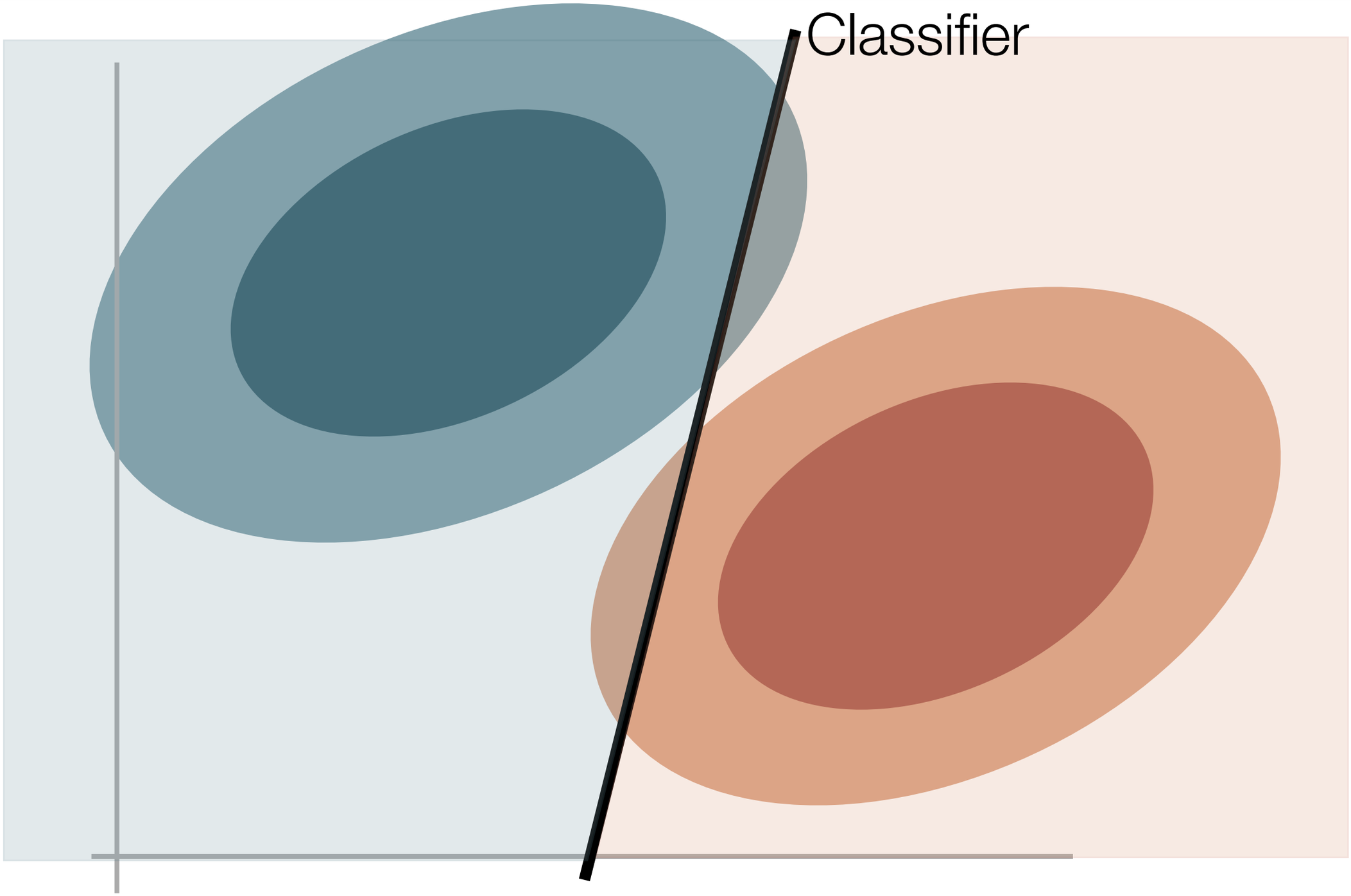
$$R(f) = P_{XY}(f(x) \neq Y)$$

- This is the part of the distribution where the input data does not match what the classifier predicts—i.e. this is the probability of the classifier being wrong

Bayes classifiers



Bayes classifiers



Bayes classifiers

- If R(f) is minimized for some f*, then f* is called a Bayes classifier

- $f^*(x) = \underset{y=1,\ldots,K}{\arg\max}\, P(x, y)$

$$f^*(x) = \underset{y=1,\ldots,K}{\arg\max}\, P(x|y)P(y)$$

$$f^*(x) = \underset{y=1,\ldots,K}{\arg\max}\, P(y|x)$$

- Many classifiers fill in these probabilities using different assumptions

Naive Bayes classifier

- Bayes classifier that assumes independence on the different input features x = (x1, …, xd) (i.e. it assumes x1, …, xd are independent)

- Strong assumption! But makes a lot of things easy

- Useful for predictors that have a finite range/categorical

- In particular, often used for text data! (like twitter data, also spam filters, etc.)

- Eager learner but relatively fast

Why is independence so useful?

- $f^*(x) = \underset{y=1,\ldots,K}{\arg\max}\, P(x|y)P(y)$

- How to calculate $P(x|y)$ when $x = (x_1, \ldots, x_d)$?

  $P(x|y) = P(x_1|x_2, \ldots, x_d, y)P(x_2|x_3, \ldots, x_d, y) \ldots P(x_d|y)$

- But because of independence, we can write:

  $P(x|y) = P(x_1|y) \ldots P(x_d|y)$

- Much simpler! We can estimate these from the training data

Naive Bayes classifier with text data

- Suppose we have a set of documents, with a subset that we have hand categorized (e.g. pro- or anti-vax, spam vs. non-spam, or by topic area, etc.)

- ‘Bag of words’ approach, the feature list is all words in all documents

- Each document is given by a vector $x = (x_1, \ldots, x_d)$, where x1, …, xd are 0 or 1 depending on if a word is present

Naive Bayes classifier with text data

- From the training data we can estimate

$$P(x|y)P(y) = P(x_1|y) \dots P(x_d|y)P(y)$$

- And choose the classifier that maxes this

Example

	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes

https://www.geeksforgeeks.org/naive-bayes-classifiers/

Example

Outlook

	Yes	No	P(yes)	P(no)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	2/5
Total	9	5	100%	100%

Temperature

	Yes	No	P(yes)	P(no)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
Total	9	5	100%	100%

Humidity

	Yes	No	P(yes)	P(no)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
Total	9	5	100%	100%

Wind

	Yes	No	P(yes)	P(no)
False	6	2	6/9	2/5
True	3	3	3/9	3/5
Total	9	5	100%	100%

Play		**P(Yes)/P(No)**
Yes	9	9/14
No	5	5/14
Total	14	100%

Example

`today = (Sunny, Hot, Normal, False)`

$$\underline{P(SunnyOutlook|Yes)P(HotTemperature|Yes)P(NormalHumidity|Yes)P(NoWind|Yes)P(Yes)}$$

$$\approx 0.0141$$

$$\underline{P(SunnyOutlook|No)P(HotTemperature|No)P(NormalHumidity|No)P(NoWind|No)P(No)}$$

$$\approx 0.0068$$

So our classifier would say that today we play golf!

Other variations

- Gaussian naive Bayes assumes Gaussian distributions for the features (often used when working with continuous variables)

- Multinomial naive Bayes uses word frequency rather than just yes/no

- And many others

- Many neural network tasks are really all about classification too—that is a whole other flavor of classifiers that we'll talk more about next time!

Neural Networks

Neural networks

- A simplified model of neurons in the brain—used both in studying neuroscience and as a machine learning tool (often sort of divorced from the the biological aspects of the model)

- **Connectionism** - an approach that aims to understand neurons and the brain by modeling the network structure of connections between neurons

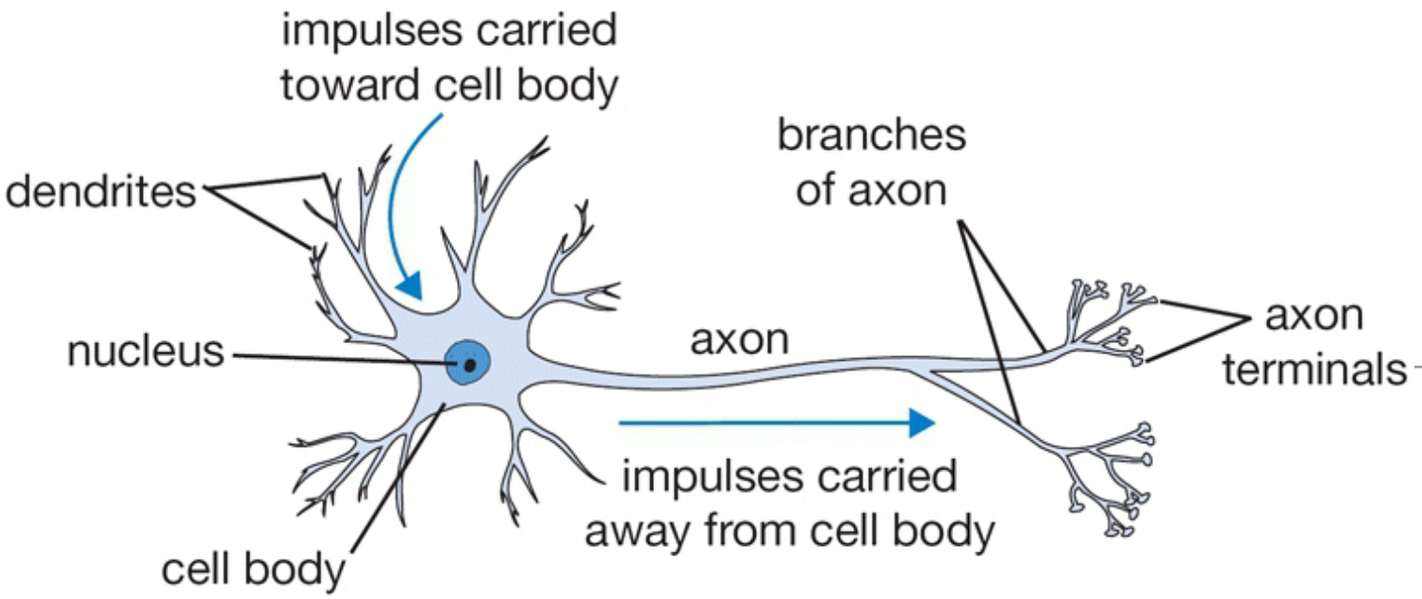
- Misses a lot! But the network is critical

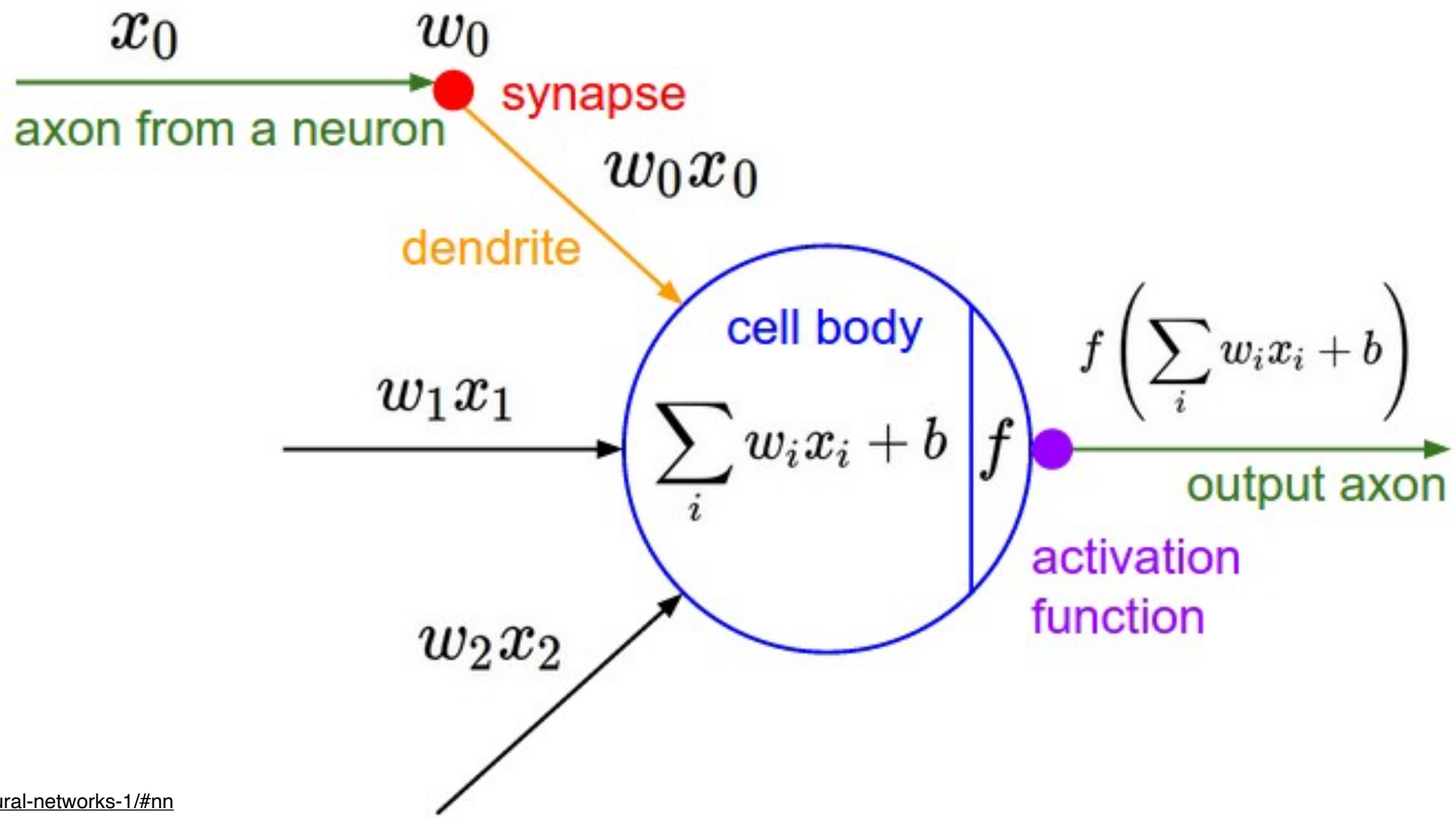
- A very complex systems approach—treat the brain as a network of agents that interact, leading to emergent phenomena of object recognition, etc.

How do neural networks work?

- **“Neurons” or units** - each agent or node in the network. Neurons in are essentially functions: they take as inputs the weighted sum of the incoming connections and return some function of that number they receive.
    - The function that they use is called the **activation function**.
- **Connections** - the edges or links from one neuron to another in the network. These are considered as directed edges
- **Weights** - how the receiving neuron weights each edge as it calculates its total inputs
- **Layers** - most neural networks are organized in layers (analogous to many brain regions)

https://cs231n.github.io/neural-networks-1/#nn



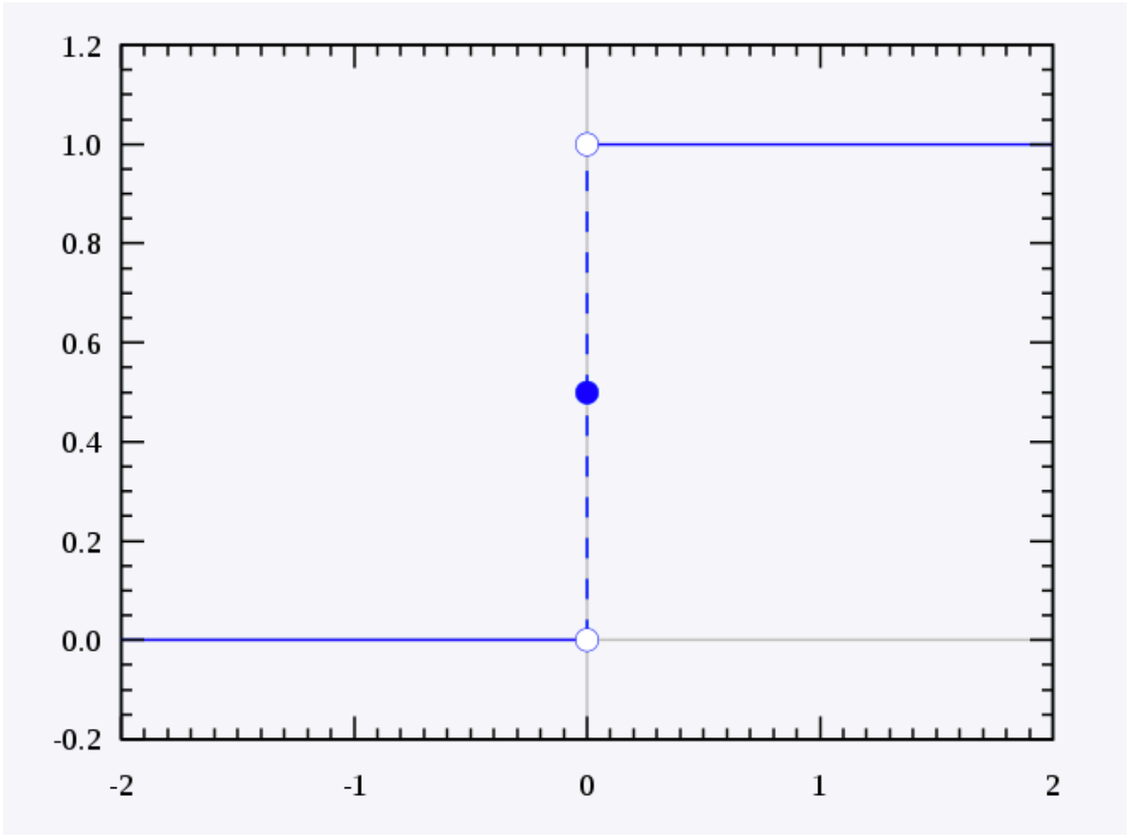


https://cs231n.github.io/neural-networks-1/#nn

Binary activation function

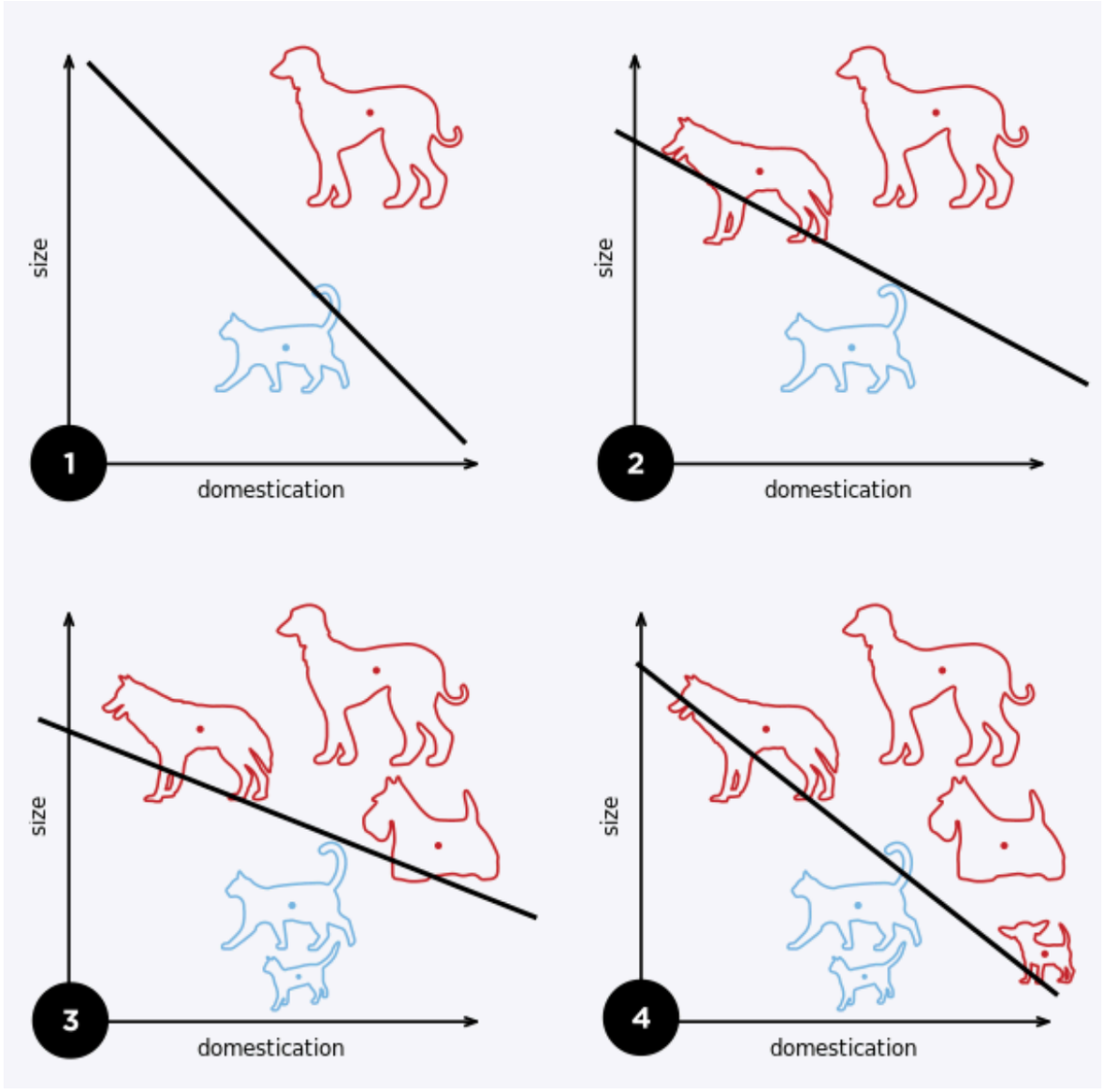
- When the weighted sum of the inputs is greater than some threshold, the neuron fires

- Mostly used for research rather than real-world AI/ML in practice now, but early neural networks used this



Perceptron

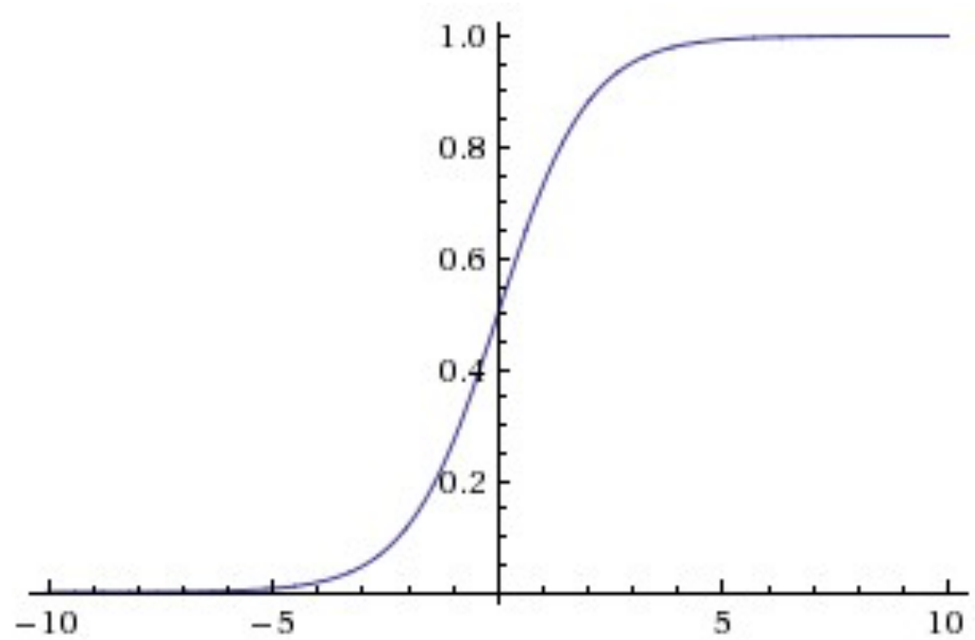
- Simple neural network model that uses a binary activation function and a single layer to classify objects or do edge detection
- Developed in 1943
- Basically does a single linear classifier
  - I.e. when the weighted sum of the inputs is greater than some threshold, it fires or doesn't

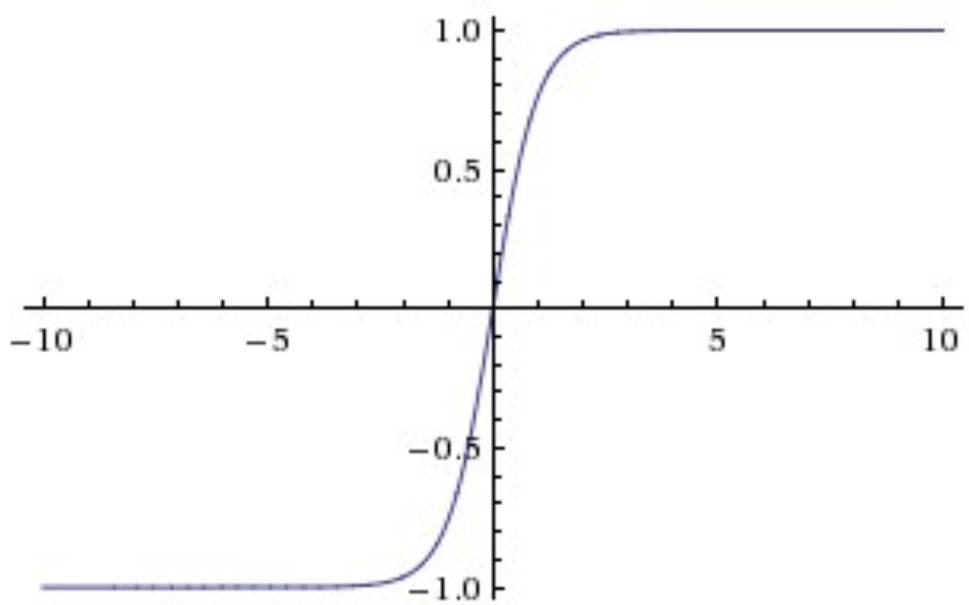


Activation functions often represent firing rate rather than binary firing

- In this case, the “neuron” should really be thought of as measuring a small patch or population of neurons

Sigmoid activation function

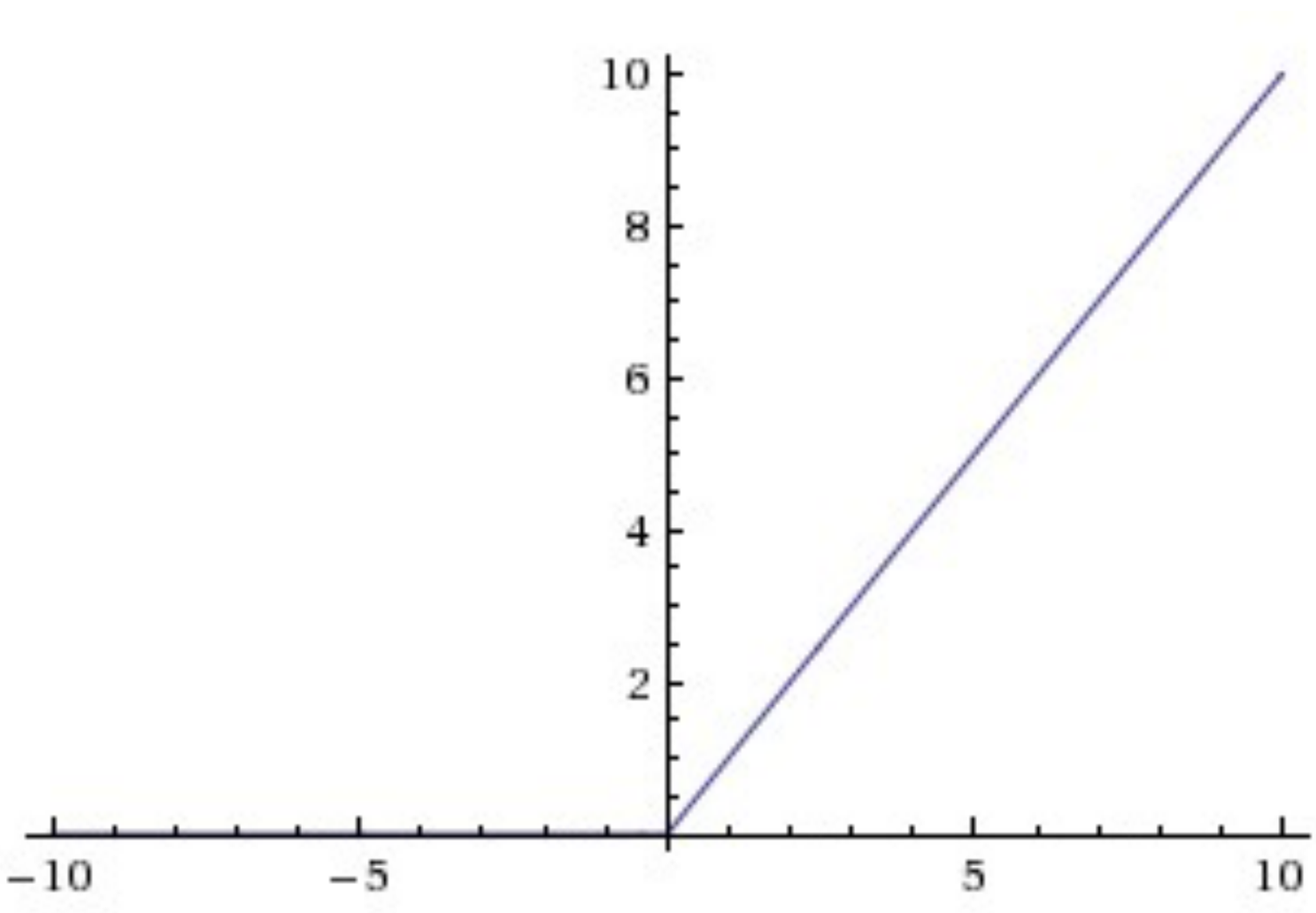




https://cs231n.github.io/neural-networks-1/#nn

Activation functions often represent firing rate rather than binary firing

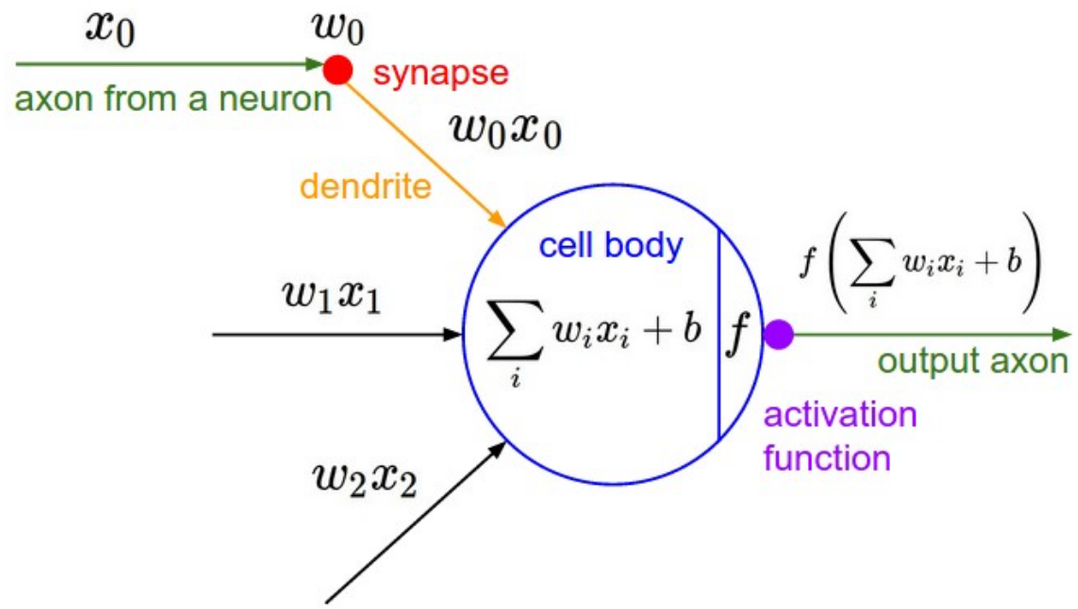
- Rectified linear unit or ReLU
- 0 until some threshold then linear

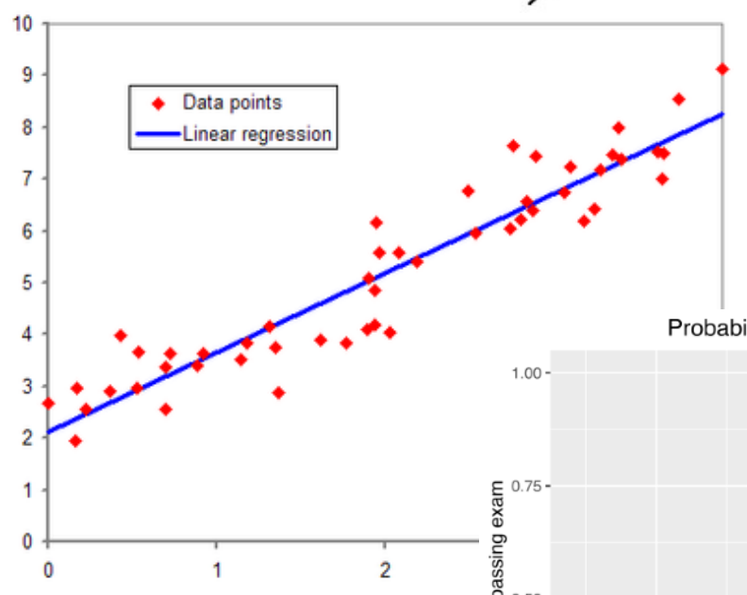


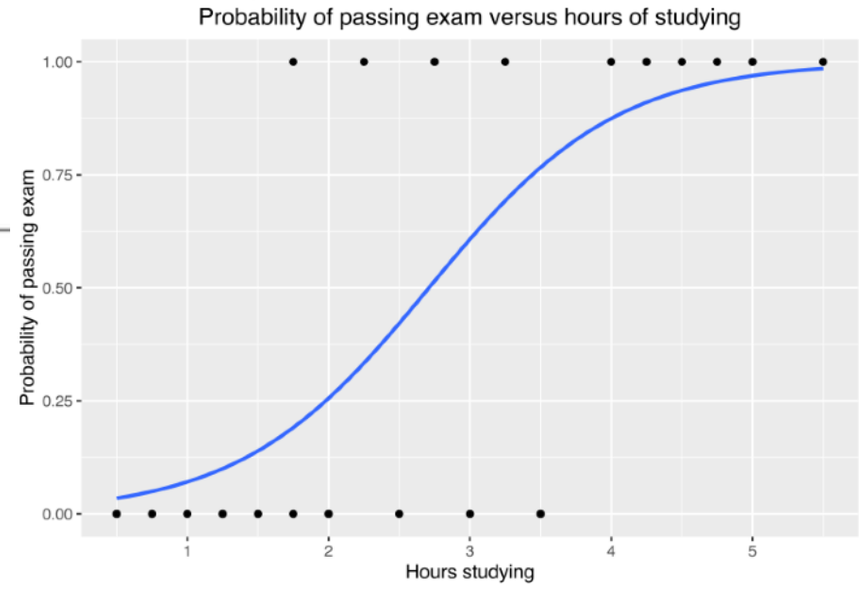
One neuron is basically linear or logistic regression (or similar regression models)

- A neural network is (in some sense) a high dimensional regression model!

- Example 1 here for instance: https://xnought.github.io/backprop-explainer/

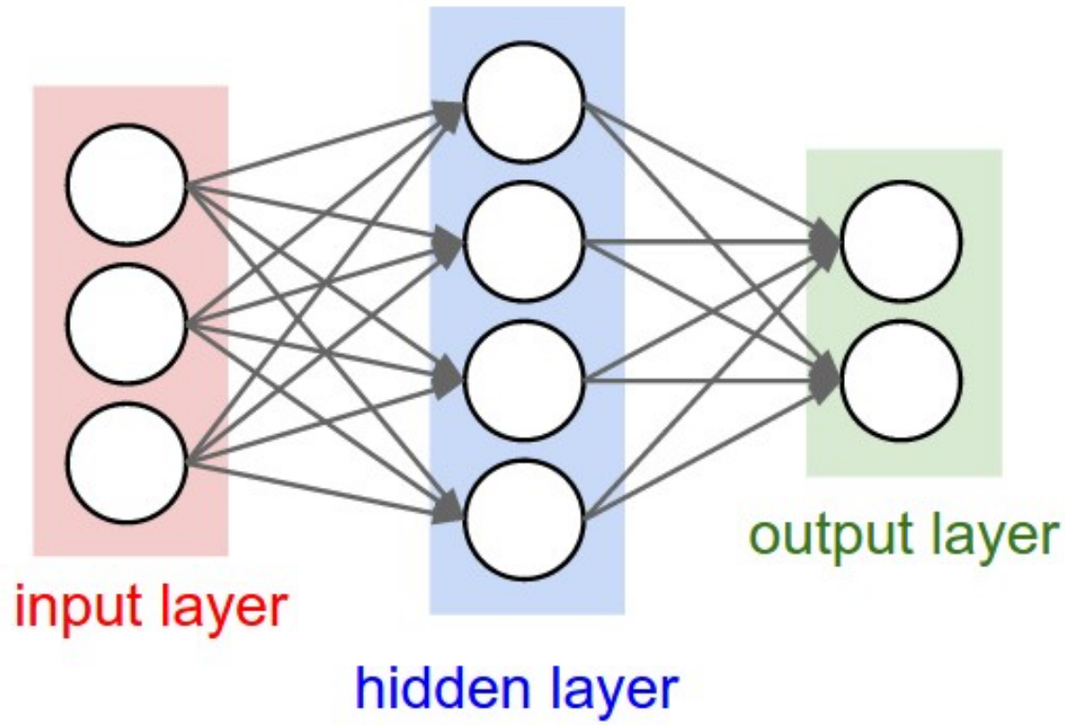






Layers of neurons allow them to do more advanced processing

- 2-layer Neural Network
  - 1 hidden layer of 4 neurons
  - 1 output layer of 2 neurons

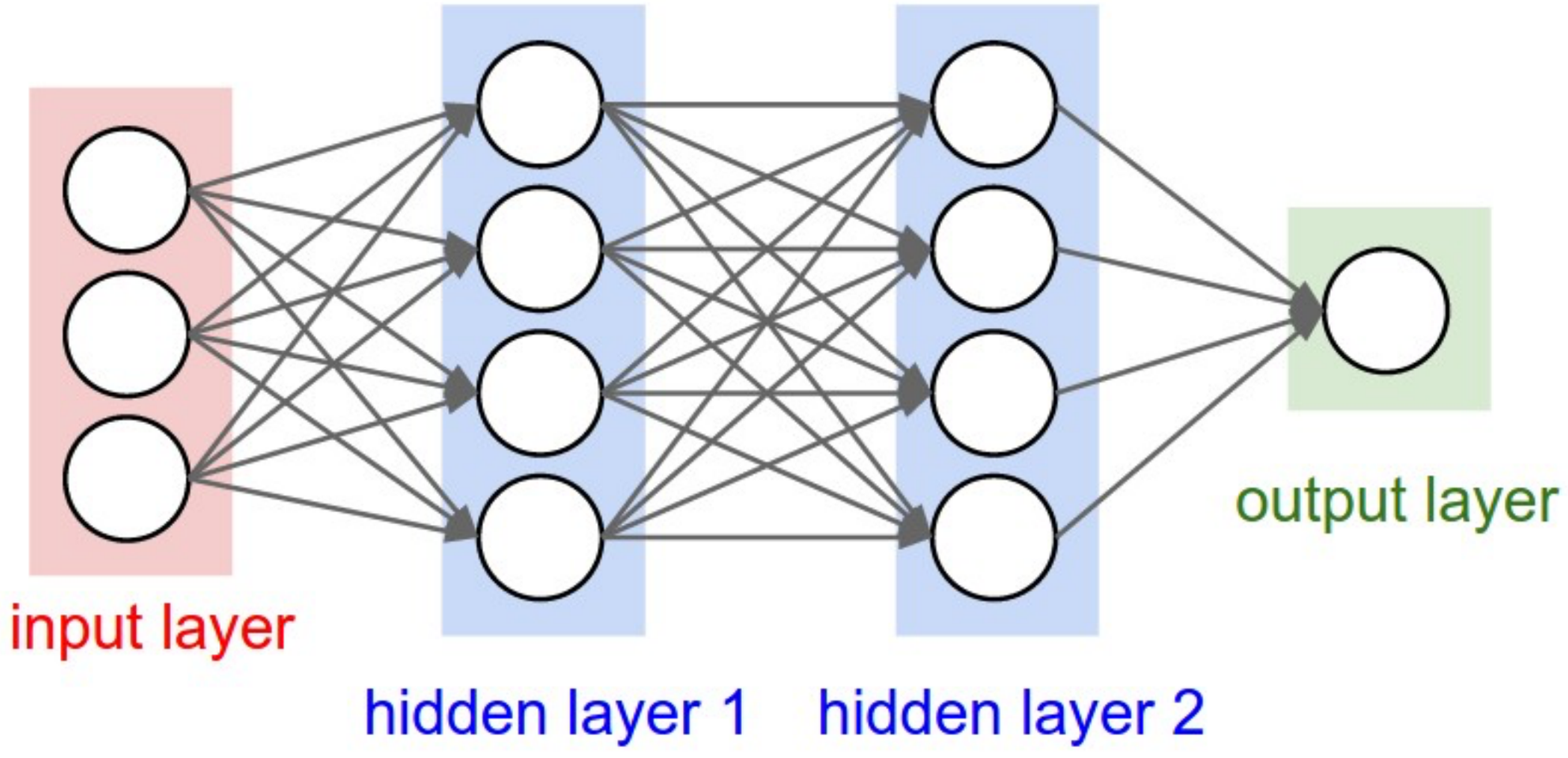


- We don’t count the input layer as this doesn’t do any processing, just represents the inputs (i.e. the data) as a set of numbers that feed into the subsequent layers

https://cs231n.github.io/neural-networks-1/#nn

Layers of neurons allow them to do more advanced processing

- 3-layer Neural Network: 2 hidden layers of 4 neurons, 1 output layer of 1 neuron



https://cs231n.github.io/neural-networks-1/#nn

What do these layers do?

- Because each neuron in the last layer feeds every neuron in the next layer (but with weights), the weighted sums are often just matrix multiplication!

- The layers act as repeated matrix operations (linear) interspersed with a nonlinear activation function

What do these layers do?

- Means we can approximate complex shapes, surfaces, classifications, etc. very closely—and we get the benefit of a huge amount of theory & methods for linear algebra

Neural networks with at least one hidden layer are universal approximators

- Neural networks with at least one hidden layer can approximate any continuous function to arbitrary precision

- (Note the classic perceptron had no hidden layer! It only does a linear classifier)

Neural networks with at least one hidden layer are universal approximators

- Most advanced neural networks (e.g. ChatGPT) are deep neural networks, meaning they have multiple (often many!) hidden layers

- Why?

- Just because something can approximate a classification/function/data set, doesn’t necessarily make it a good ML/AI algorithm in practice (for example, connecting the dots also approximates well!)

Deep neural networks

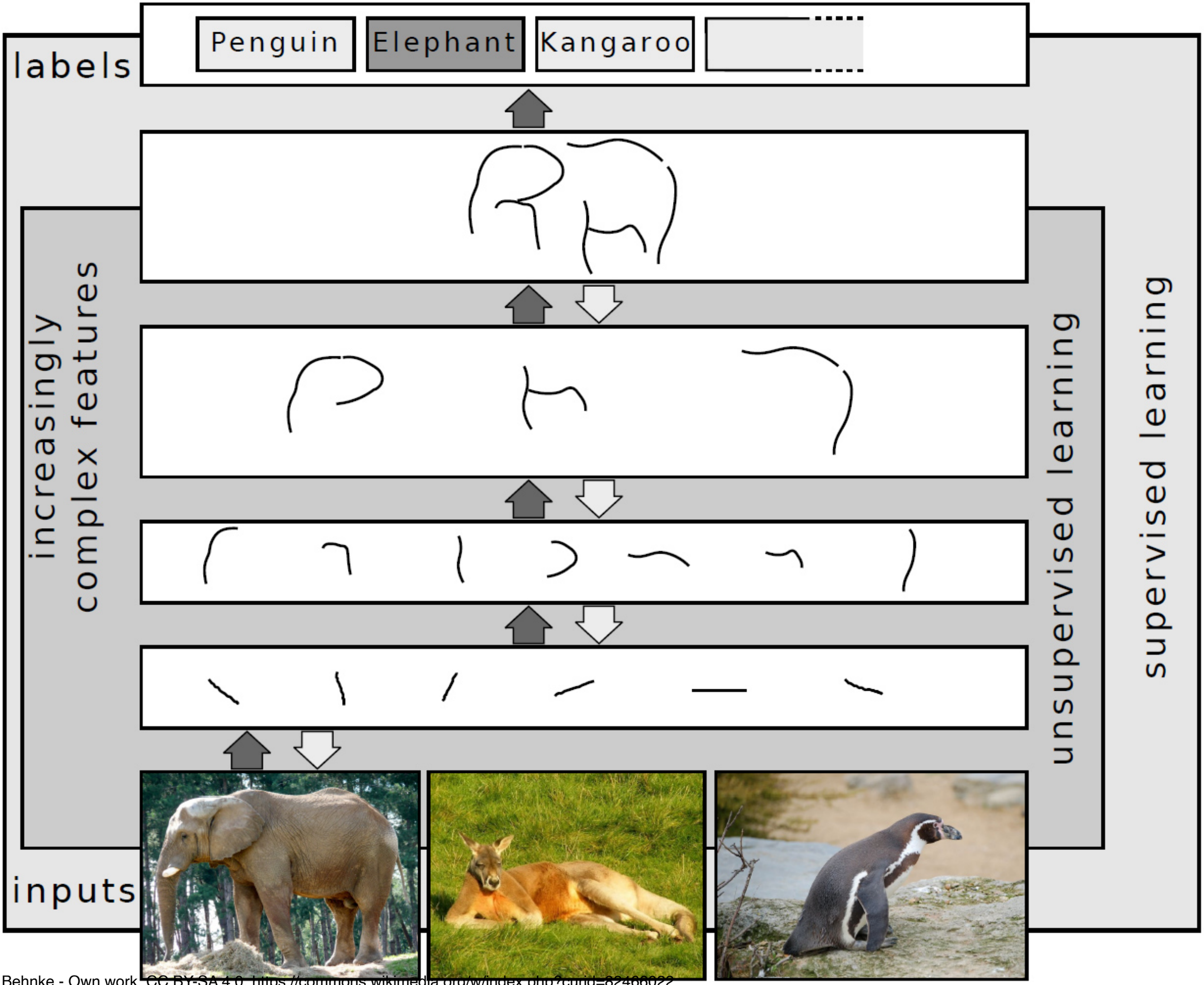
- Neural networks with multiple hidden layers
- Although networks with one hidden layer can theoretically perform the same representation of the data as deep neural networks, often deep neural networks outperform shallow neural networks in practice
- Why?

Deep neural networks

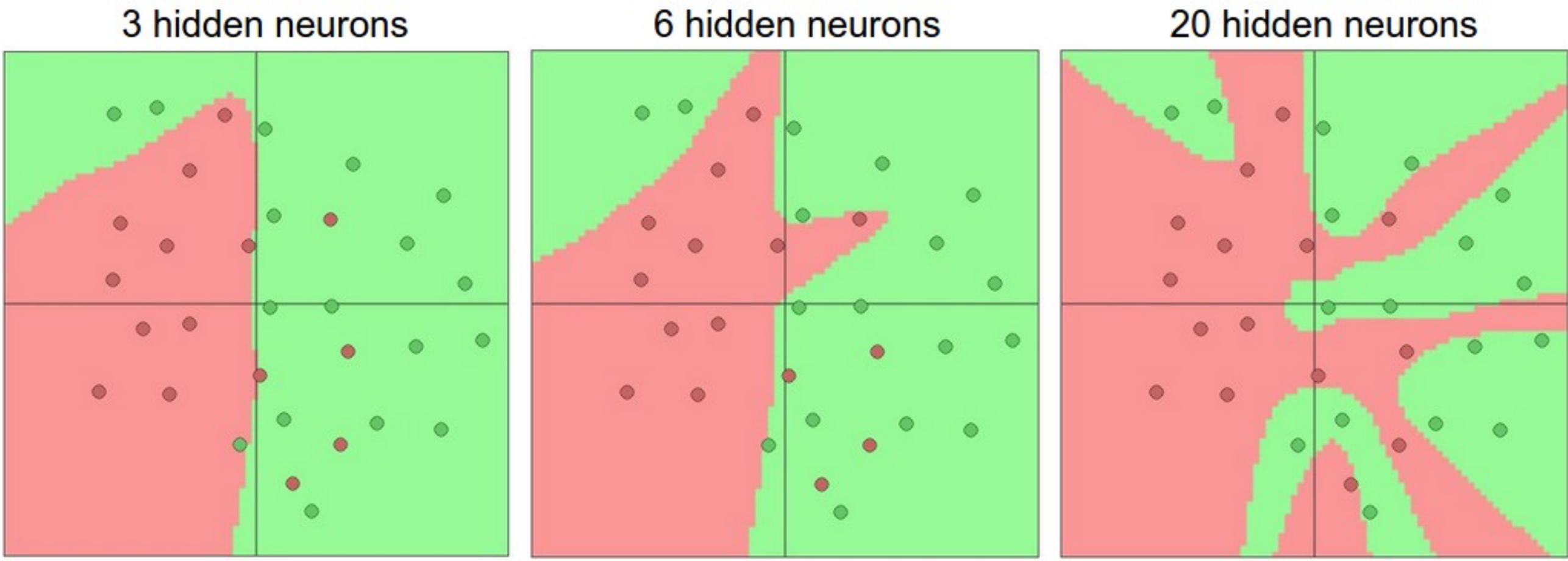
- Data often contains hierarchical structure (e.g. pixels make up edges which define objects and so on, or cancer subtypes, etc.), so several layers of processing allow us to capture higher order features that would be more difficult to capture purely from the initial input data alone

Deep neural networks

- E.g. the pixels in an image classifier can be processed to edge detection in the first layer, then edges can be related to each other to recognize boxes, tables, chairs, etc. in subsequent layers (eventually capturing higher level structures etc). Even though actually figuring out how the network is doing this can be quite difficult for large networks!



More hidden neurons (and more layers) improves the model's ability to approximate the data

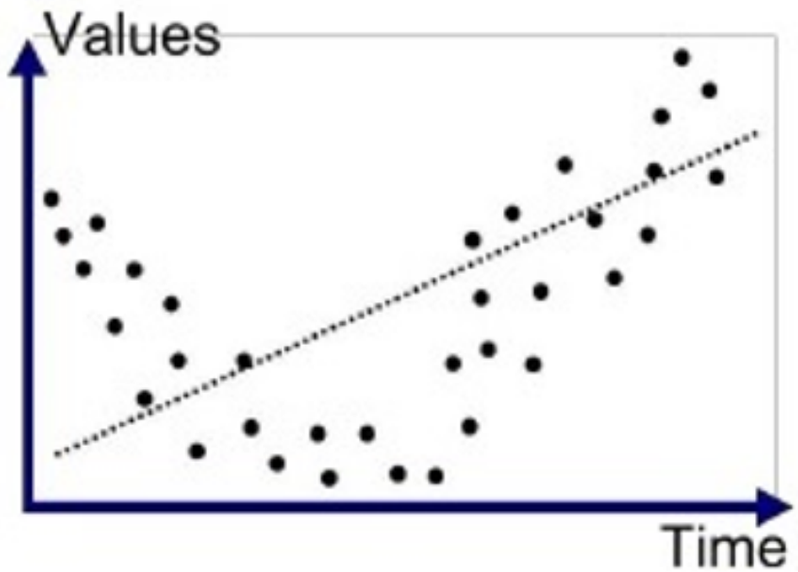


https://cs231n.github.io/neural-networks-1/#nn

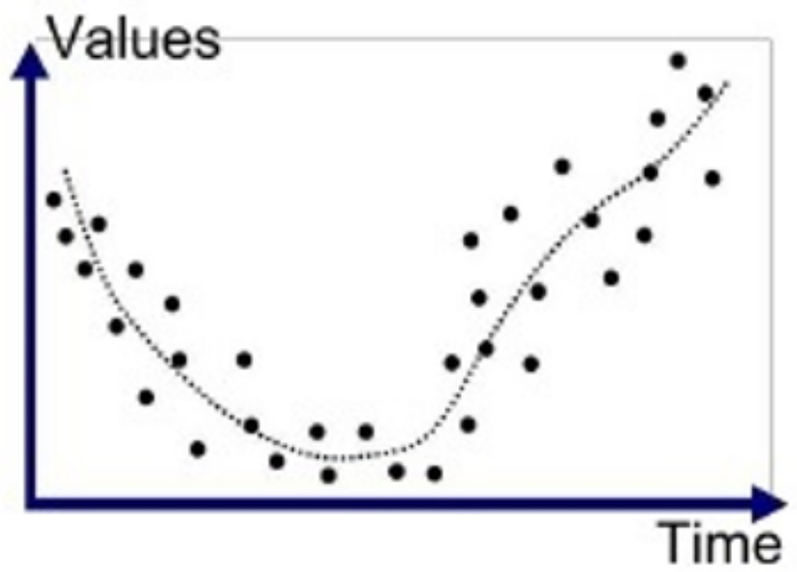
Fun example to play with

- https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

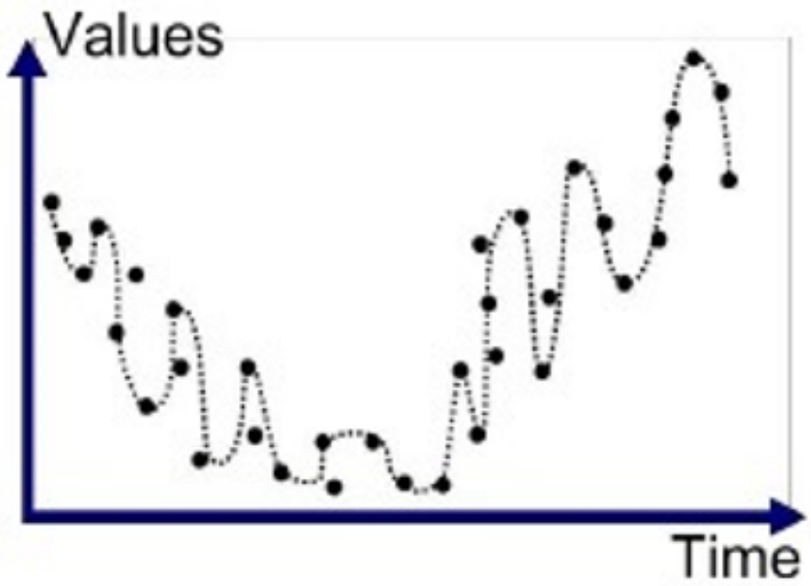
Be careful about overfitting



Underfitted



Good Fit/Robust



Overfitted

https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76

How do we train (i.e. do parameter estimation) neural networks?

- Often via gradient descent! (with all the pros and cons that come along with that)

- One of the most common approaches is via an algorithm called backpropagation—this takes advantage of the idea that to calculate the gradient on the goodness of fit surface, we can use the chain rule

- This means that we can simulate forward and then adjust the model weights moving backward from the cost function calculated at the output

Backpropagation tutorial

- https://xnought.github.io/backprop-explainer/

- But as you get to large numbers of neurons (e.g. 1000's), the surface gets much more complicated:

- https://twitter.com/jaschasd/status/1756930242965606582?s=20

- We are unlikely to ever be in the "true" global minimum (if there even is a unique one)—but do we care?

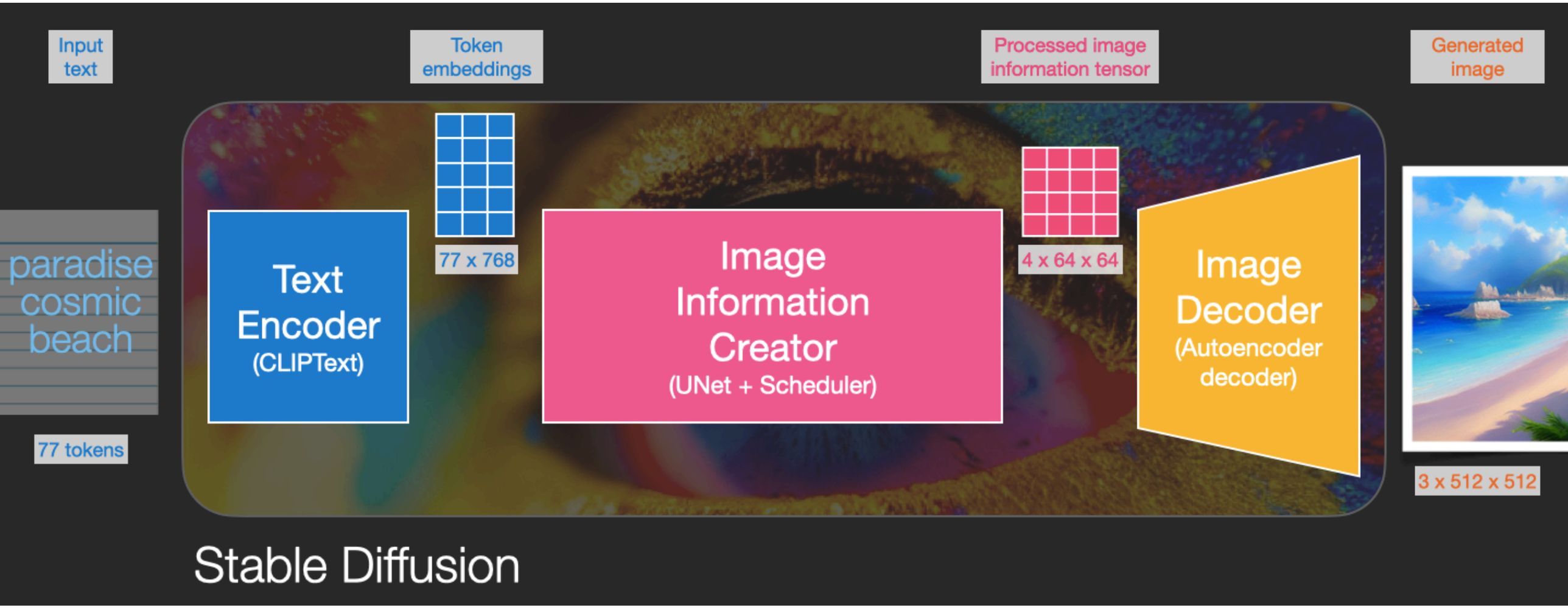
Brief intro to deep neural networks

Deep neural networks for image generation

- Stable Diffusion
- DALL-E
- Midjourney
- UM GPT has openjourney: http://umgpt.umich.edu
- Dream.ai

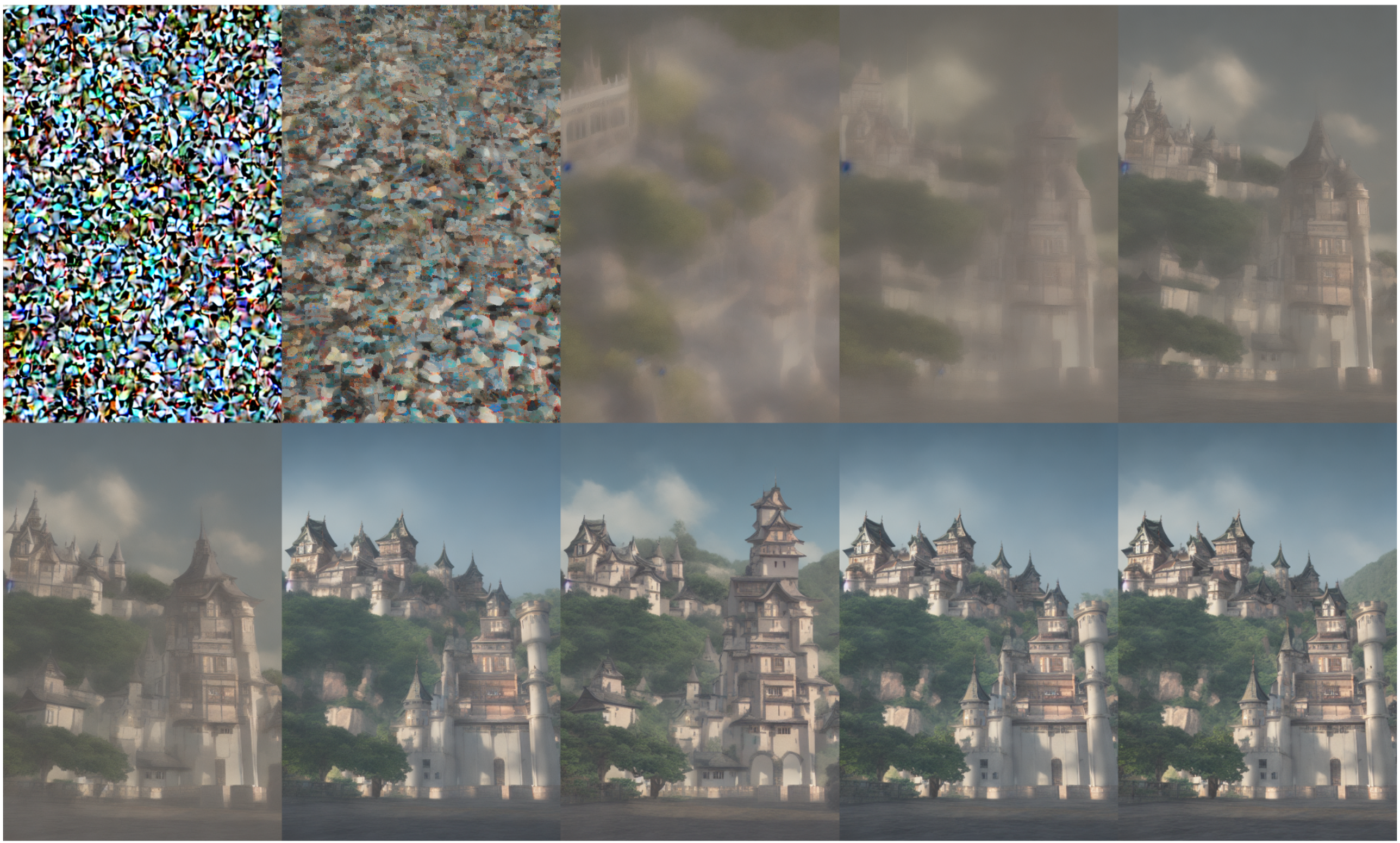
- Most of these use diffusion models—a kind of deep learning neural network approach

Stable diffusion



http://jalammar.github.io/illustrated-stable-diffusion/

Steps: 1 Steps: 2 Steps: 3 Steps: 5 Steps: 8



Steps: 10 Steps: 15 Steps: 20 Steps: 30 Steps: 40

Deep neural networks for text processing

- ChatGPT (UM's version: http://umgpt.umich.edu)
- Llama
- Bard

- These mostly rely on transformer models

More next time! Switch slide decks for now if I get here