NYU TANDON SCHOOL OF ENGINEERING

AI4CE

# Robot Perception

## Classification

Dr. Felix Juefei Xu

juefei.xu@nyu.edu

ROB-GY 6203, Fall 2022

# Overview

++ Image representation, decision boundary, and metrics

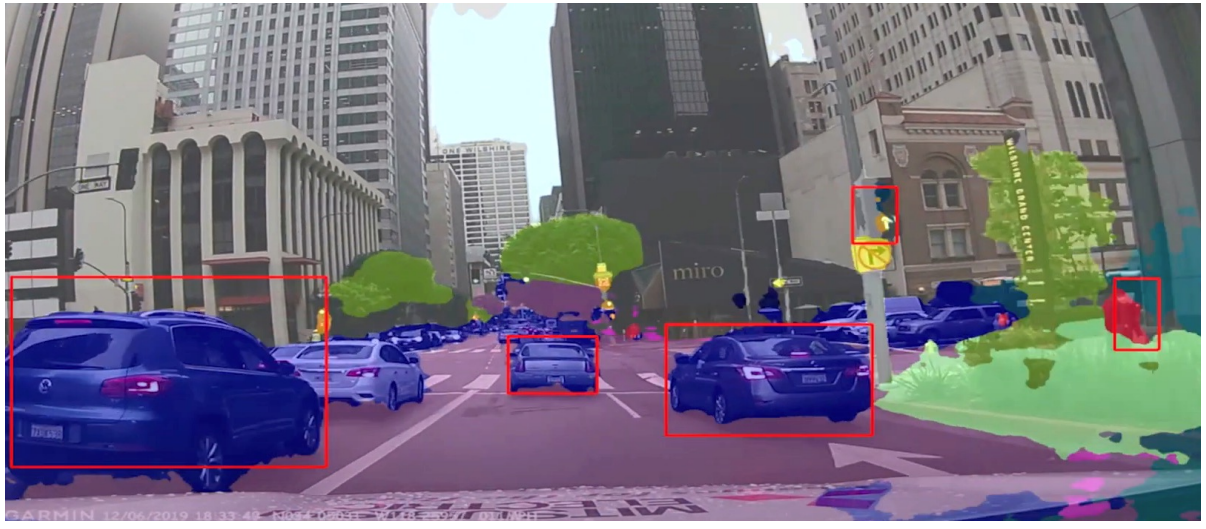
* Dimensionality reduction with PCA

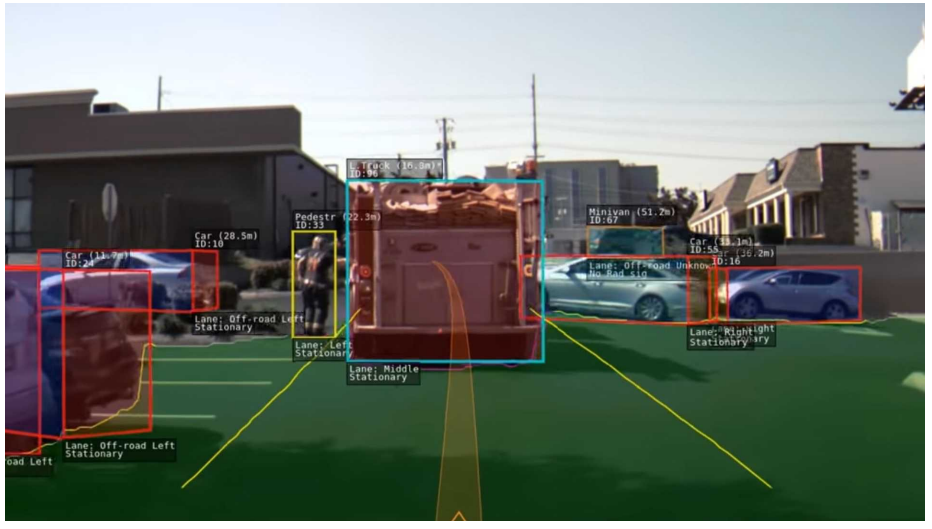
++ Building classifier
+ Supervised: SVM
++ Unsupervised: K-means

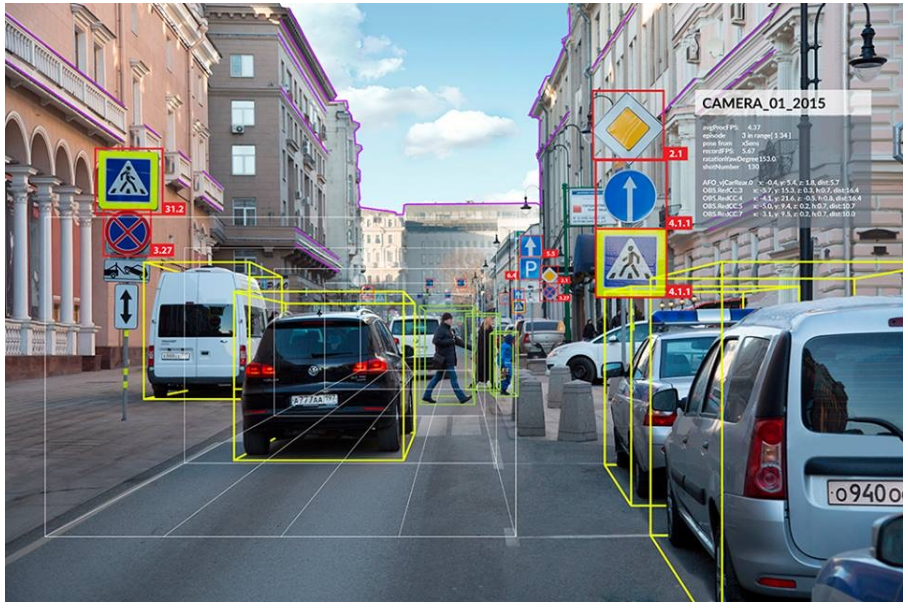
+ Deep learning based image classification
+ Advanced CNNs: MobileNet
+ Vision Transformers (ViT)

*: know how to code
++: know how to derive
+: know the concept

# Autonomous Driving Perception

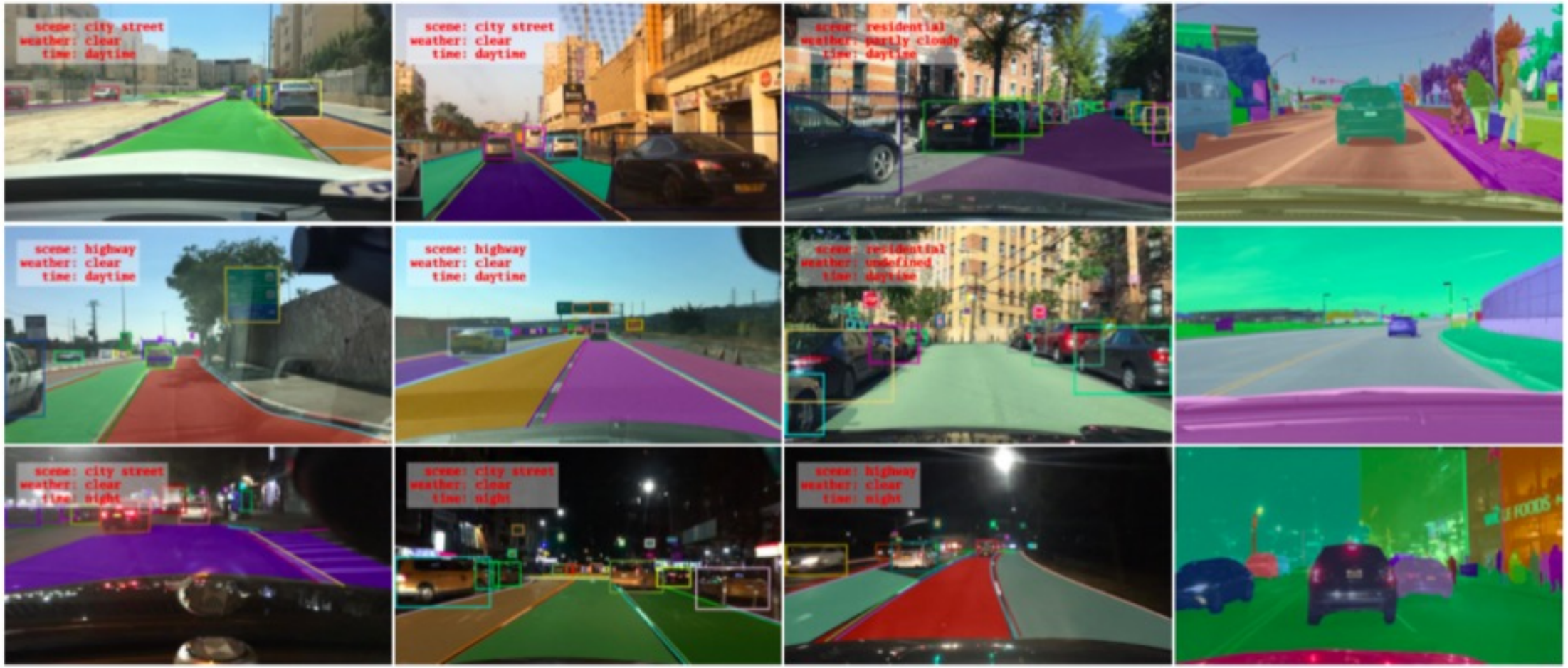




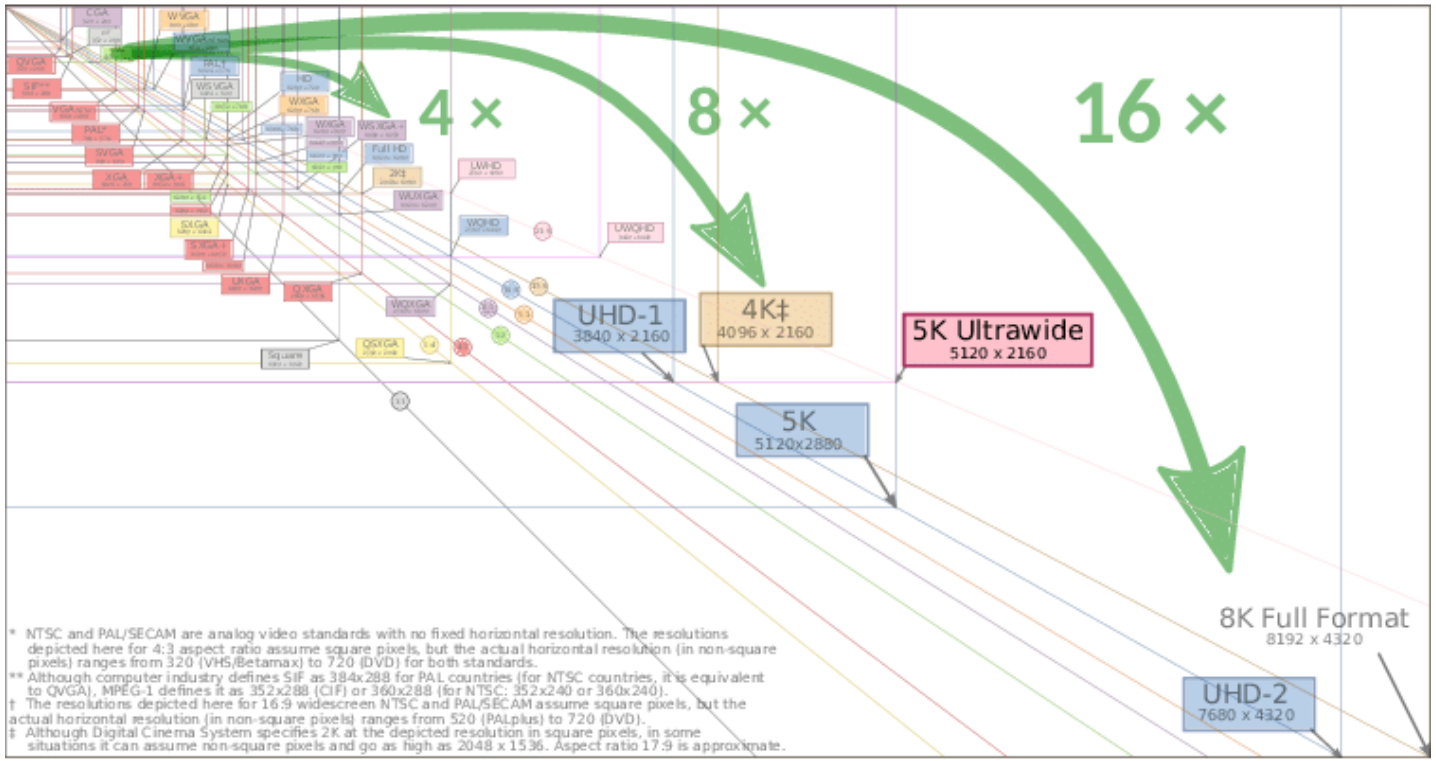

# BDD-100K: Self-Driving Perception Dataset

- Classification | Detection | Tracking | Segmentation …



# High-Dimensional Representations for Images

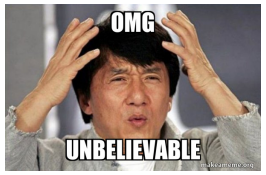
- Both sensor and display resolutions are getting higher and higher over the years


iPhone 14 Pro Max has camera resolution of 48MP
That is: 8064 x 6048 = 48,771,072

Considering RGB channels, total image dimension:

48,771,072 x 3 = 146,313,216



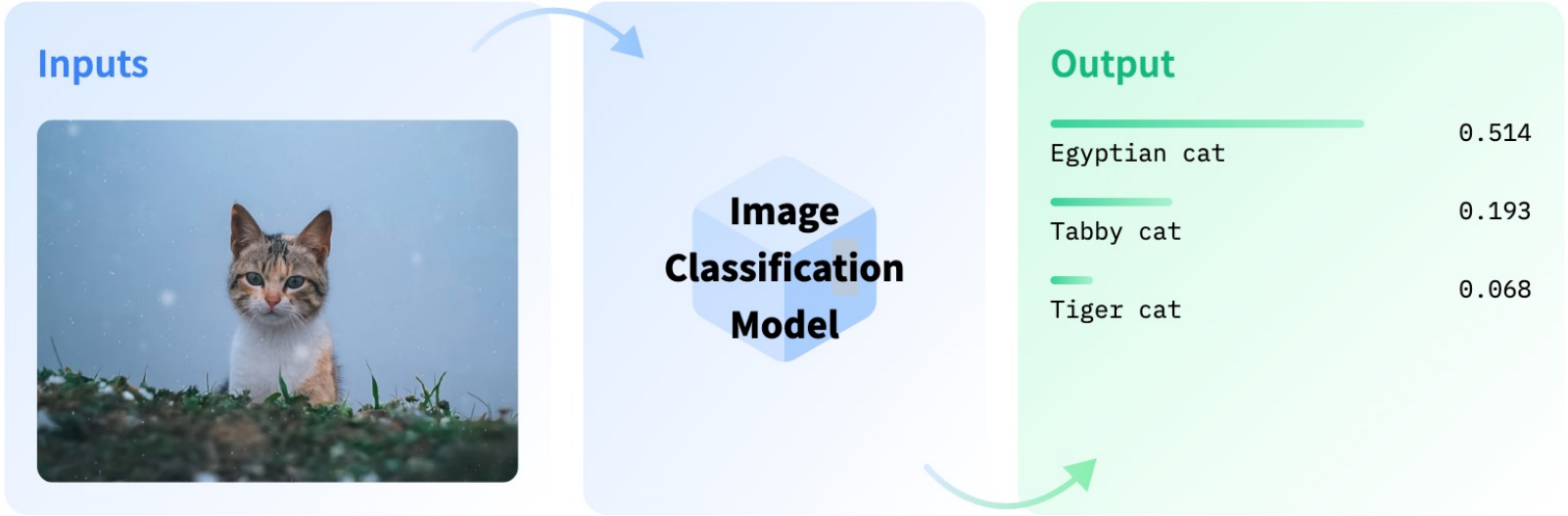
# High-Dimensional Representations for Images

- May not be a good idea to directly use pixel intensity as image representations.
  - Too high-dimensional (requires more compute, more storage / memory, etc.)
  - Too redundant (high correlation between spatially neighboring pixels)
  - Too sensitive to image geometric and photometric distortion (not robust!)
  - Too … boring … 😂

- Obtaining (Compact) Representations
  - Hand-crafted:
    - SIFT/ SURF/ DAISY/ HOG/ LBP / VLAD representations, etc.
  - Learnable
    - Shallow:
      - PCA representation
    - Deep (usually take the penultimate layer feature):
      - CNN representation
      - ViT representation

# High-Dimensional Representations for Images

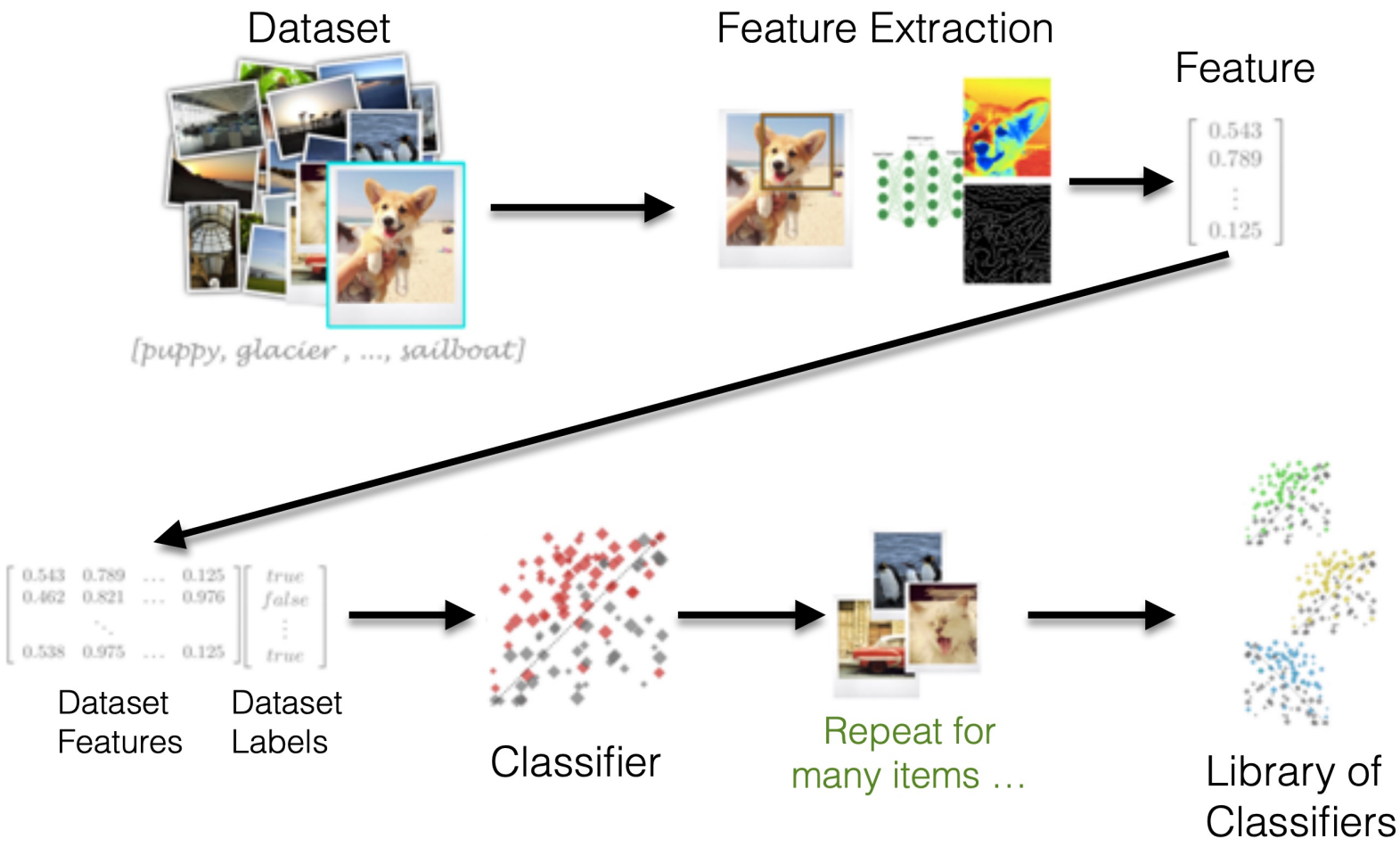
- What is (Image) Classification?
  - Is the task of assigning a (categorical) label or class to an entire image.
  - Images are expected to have only one class for each image.
  - Image classification models an image as input and return a prediction about which class the image belongs to.




https://huggingface.co/tasks/image-classification

# High-Dimensional Representations for Images

- Traditional Image Classification Pipeline



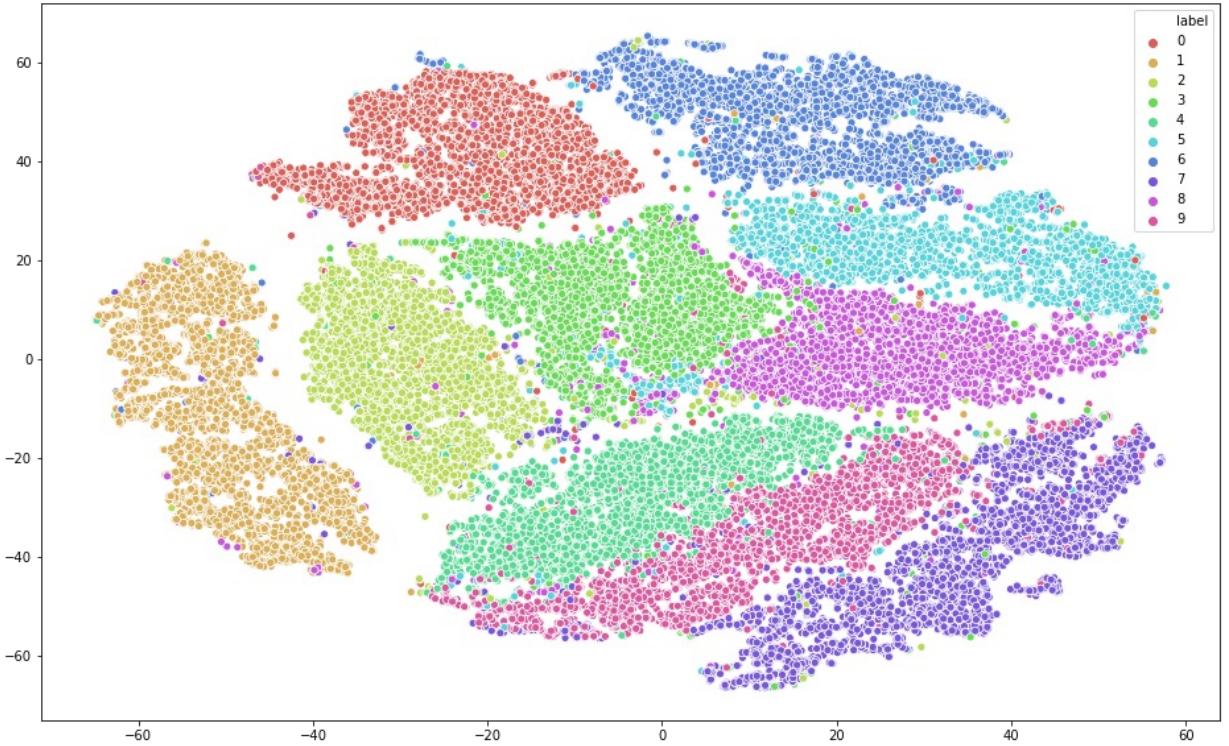
# High-Dimensional Representations for Images

- Representing an Image with a *d*-Dimensional Vector *x*
  - Think of it as a point in a *d*-dimensional space

$$\mathbf{x} \in \mathbb{R}^d$$

# High-Dimensional Representations for Images

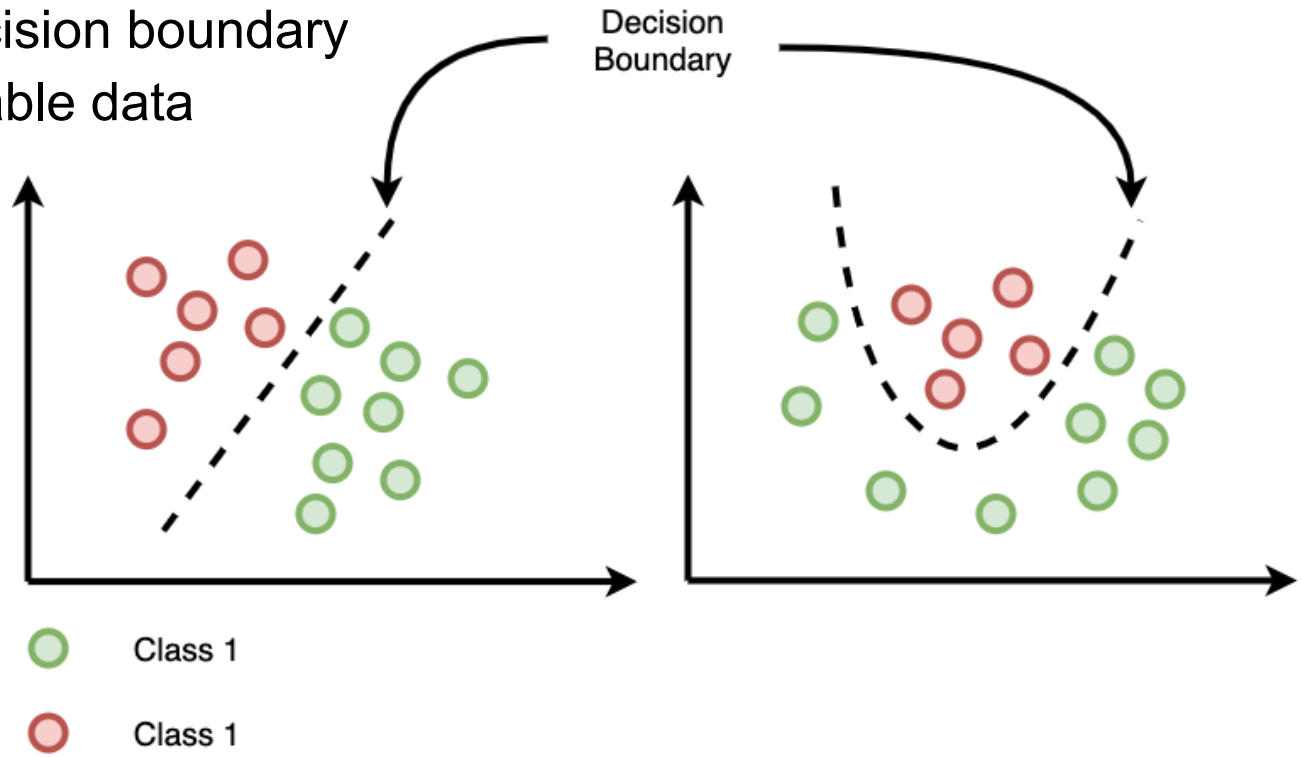
- Representing an Image with a *d*-Dimensional Vector *x*
  - Think of it as a point in a *d*-dimensional space
  - We can only visualize 2D or 3D representations
    - E.g., 2D Scatter plot of MNIST data after applying PCA (n_components = 50) and then t-SNE




https://towardsdatascience.com/dimensionality-reduction-using-t-distributed-stochastic-neighbor-embedding-t-sne-on-the-mnist-9d36a3dd4521

# Decision Boundary

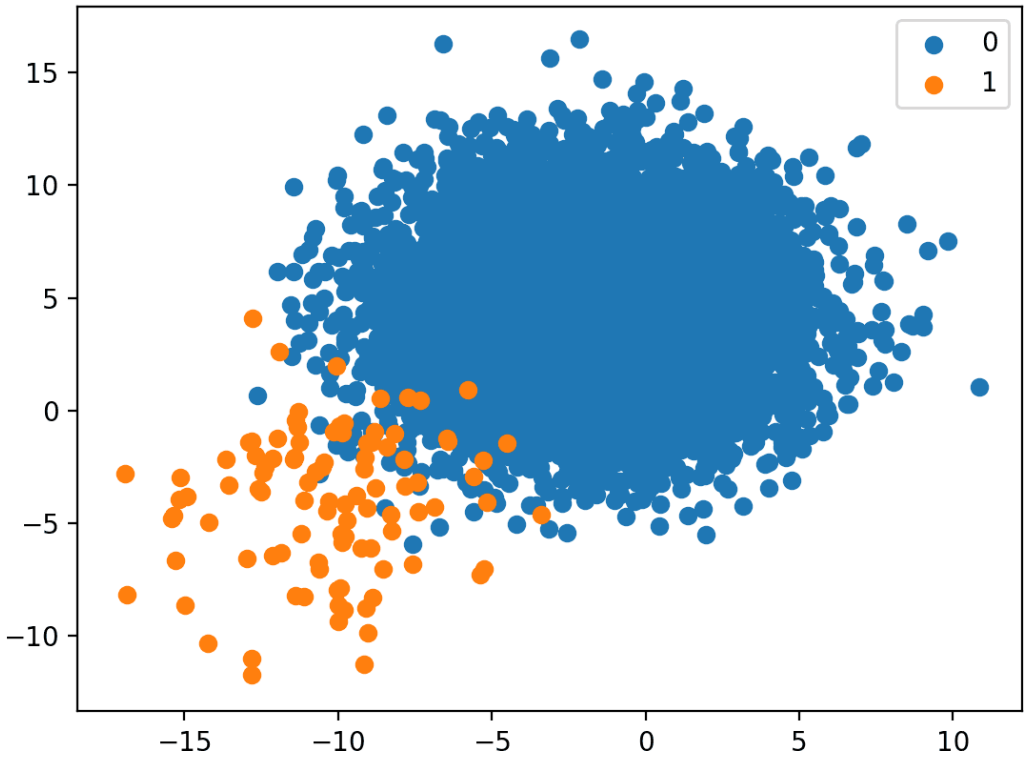
- Based on the observed instances (with labels), the goal of classification is find/ fit to a decision boundary so that we will be able to predict which class a new data instance (feature vector/ representation) might correspond to.
  - Binary vs. multi-class
  - Linear, piecewise linear, vs. nonlinear decision boundary
  - Linearly separable vs. non-linearly separable data
  - Kernel trick *(non-linearly separable data becomes linearly separable when mapped to higher dimensional representation)*
    - E.g., kernel SVM




https://towardsdatascience.com/logistic-regression-and-decision-boundary-eab6e00c1e8

# Metrics

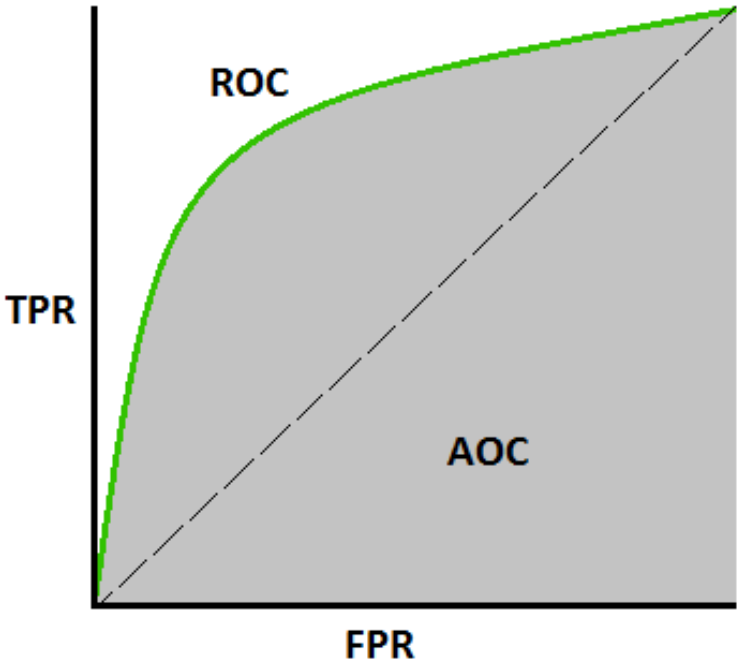
- Does “Accuracy” Suffice?
  - Accuracy = Correct Predictions / Total Predictions
  - Error Rate = Incorrect Predictions / Total Predictions
  - Accuracy = 1 – Error Rate
- Issue with Imbalanced Data
  - Class A : Class B = 1:100 class imbalance
  - If classifier predicts everything as Class B
    - Acc = 100 / 101 ~= 99.01%
- How about Per-class Accuracy?
  - (Much) better than total accuracy
  - But not compact …

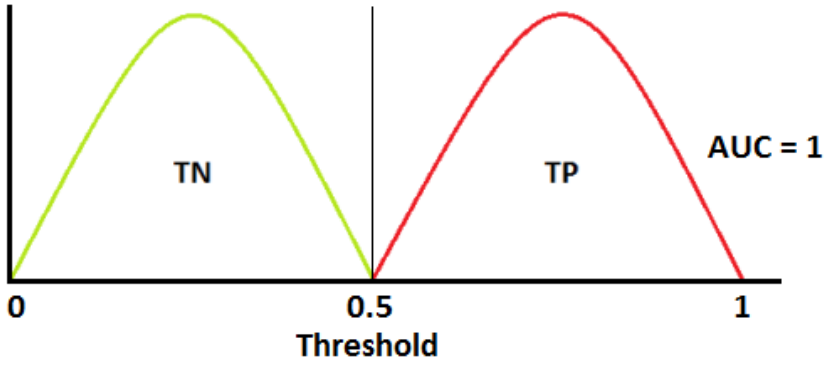


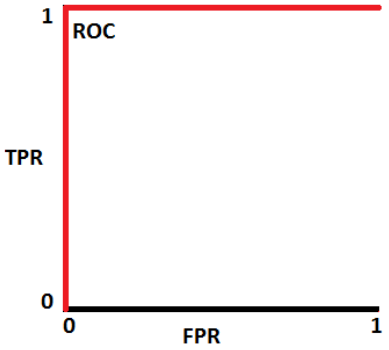

https://machinelearningmastery.com/failure-of-accuracy-for-imbalanced-class-distributions/

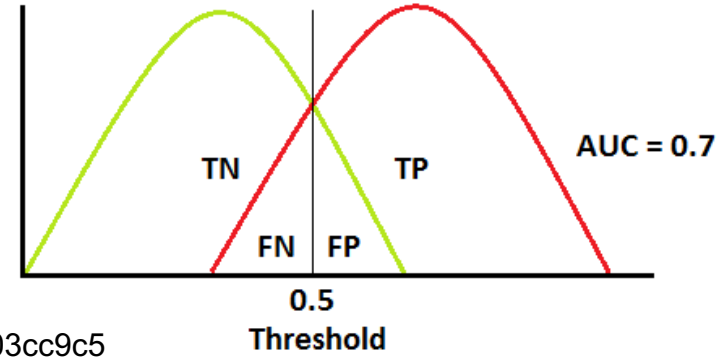
# Metrics

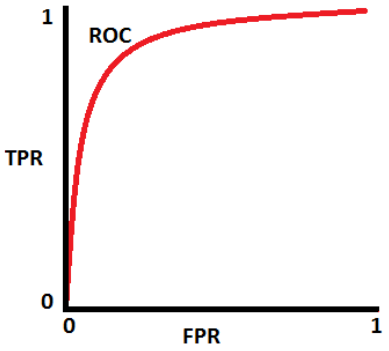
- Understanding Receiver Operating Characteristics (ROC) Curves
  - True positive rate (TPR) / recall / sensitivity = TP / (TP + FN)
    - Specificity = TN / (TN + FP)
  - False positive rate (FPR) = 1 – Specificity = FP / (TN + FP)



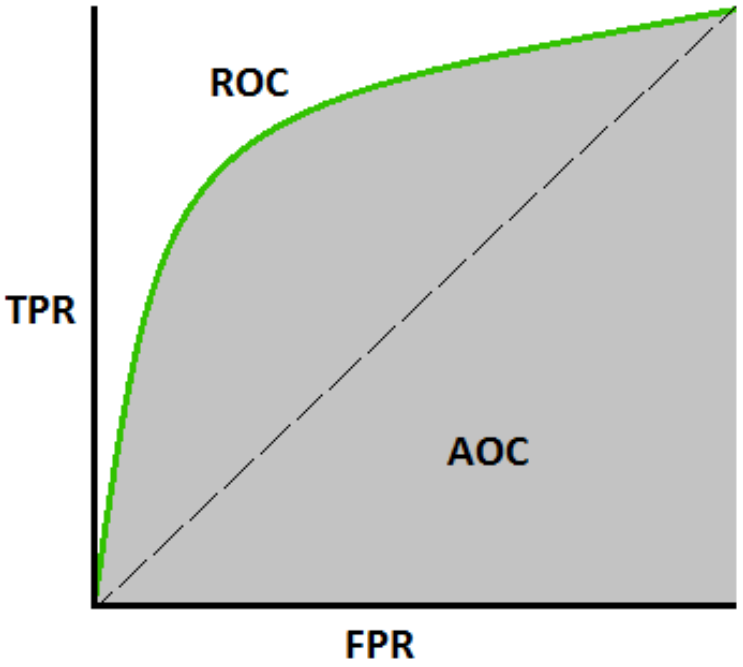



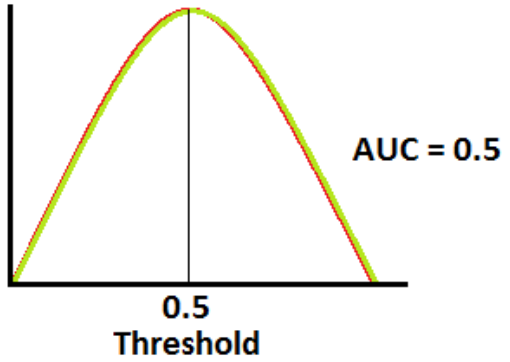



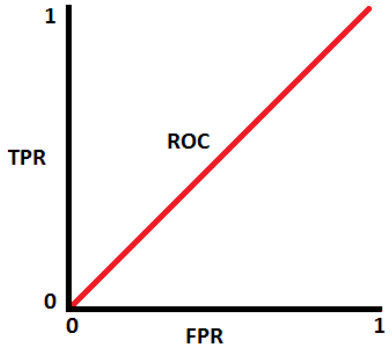

https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

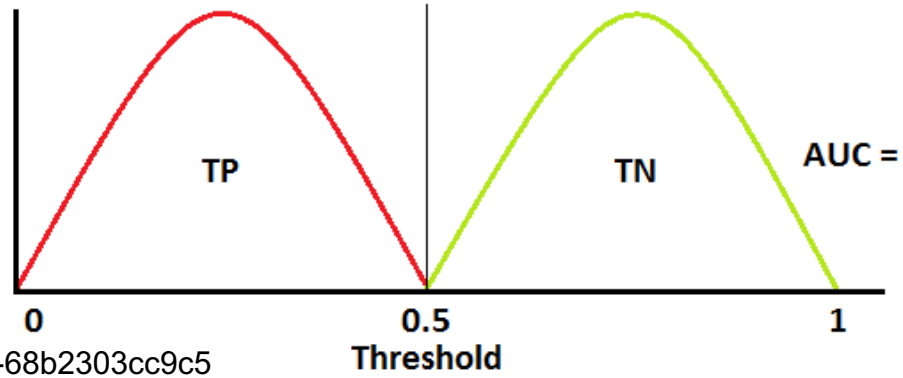
# Metrics

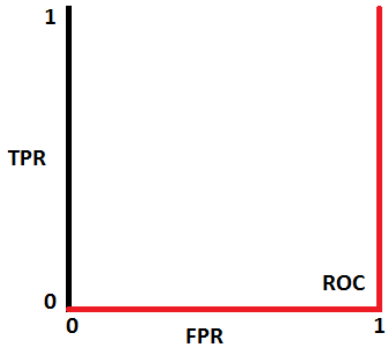
- Understanding Receiver Operating Characteristics (ROC) Curves
  - True positive rate (TPR) / recall / sensitivity = TP / (TP + FN)
    - Specificity = TN / (TN + FP)
  - False positive rate (FPR) = 1 – Specificity = FP / (TN + FP)




https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

# Dimensionality Reduction with PCA

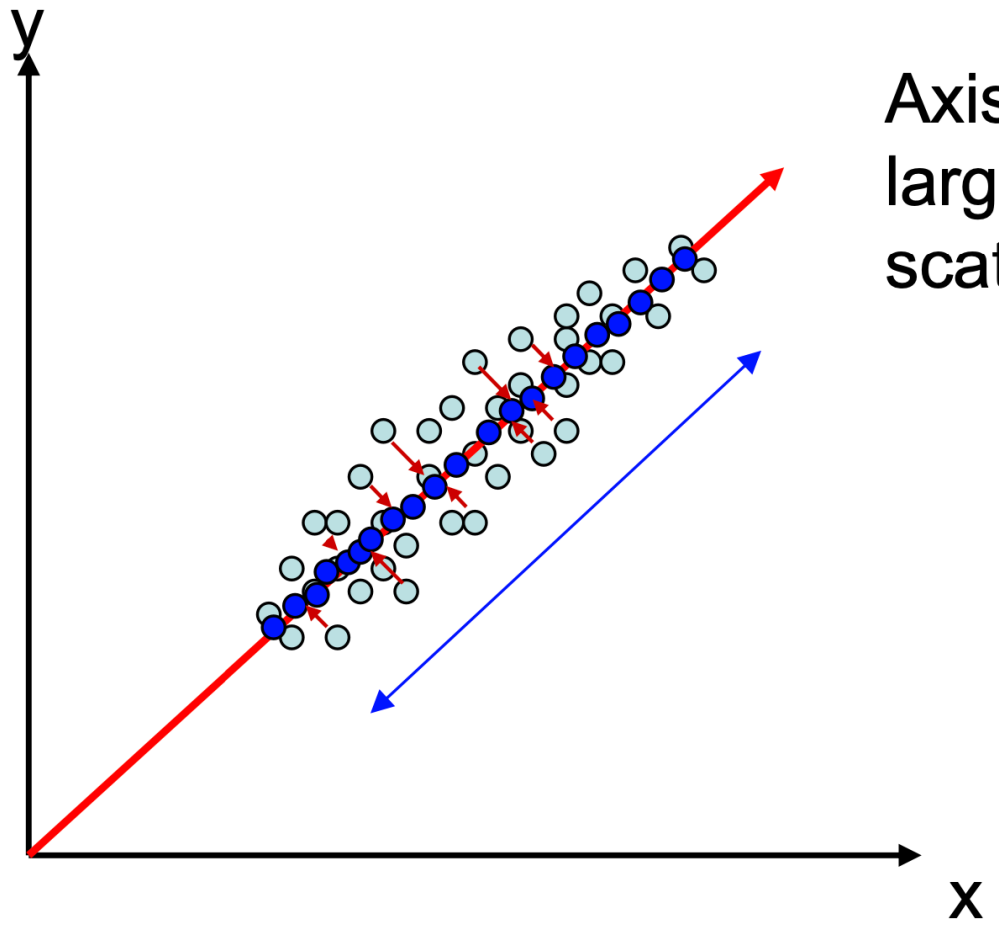
# Dimensionality Reduction with PCA

- Principal component analysis (PCA), also known as Hotelling Transform or Karhunen Loeve Transform
- Very popular pattern recognition tool for dimensionality reduction
- Commonly used baseline benchmark when testing your new pattern recognition algorithm
  - I.e., given same training data and testing configuration, can I do better than PCA

- How would you reduce the dimensionality of a data representation vector?

# What is PCA? What Are We Trying To Do?

- We want to find projections of data (i.e., direction vectors that we can project the data onto) that describe the maximum variation.




Axis that describes the largest variation (or scatter)

# PCA Formulation

- We want projection vector $\omega$ that when we project the data onto these directions, we maximize the scatter or variance.

- We want to maximize $\text{Var}(\omega^T x)$

  - Subject to the constraint that $\omega$ is a direction vector of unit norm, i.e., $||\omega|| = 1$

# How is PCA Derived?

- Maximizing the following objective function:

$$
\begin{aligned}
J(\omega) &= \mathrm{Var}(\omega^T x) \\
&= E(\omega^T x - \omega^T \mu)^2 \\
&= E(\omega^T x - \omega^T \mu)(x^T \omega - \mu^T \omega) \\
&= \omega^T E(x - \mu)(x - \mu)^T \omega \\
&= \omega^T \Sigma \omega
\end{aligned}
$$

- Where the covariance matrix $\Sigma = E(x - \mu)(x - \mu)^T$

# PCA Derivation

- We want to maximize J(ω) subject to the projection vectors ω being unit norm.
  - i.e., maximize quadratic term $\omega^T \Sigma \omega$ subject to ||ω|| = 1

- Solution:
  - Form Lagrangian optimization to take of constraints, take derivative and set to zero to find ω vectors.

Solve: $L(\omega, \lambda) = \omega^T \Sigma \omega - \lambda(\omega^T \omega - 1)$

take derivative w.r.t. ω, set it to zero

$$\frac{\partial L(\omega, \lambda)}{\partial \omega} = 2\Sigma\omega - 2\lambda\omega = 0 \quad \Rightarrow \quad \Sigma\omega = \lambda\omega$$

Standard Eigenvalue / Eigenvector problem (Ax = λx), i.e., the vectors ω are the eigenvectors of the covariance matrix Σ

# PCA Derivation

- Let's revisit our original objective of PCA
- We want to find projections which describe the biggest variance (or have the maximum scatter of data)
- Remember the variance of projected data is: $\omega^T \Sigma \omega$ (1)

- And our solution yielded: $\Sigma\omega = \lambda\omega$ (2)

Plug (2) in (1) and we get:

$$\begin{aligned} \text{projected variance} &= \omega^T \Sigma \omega = \omega^T \lambda\omega \\ &= \lambda\, \omega^T \omega \qquad \text{(remember } \|\omega\|=1\text{)} \\ &= \lambda \end{aligned}$$

# PCA Derivation

- This means that the direction vector $\omega_i$ has captured $\lambda_i$ variance.
- So, we want to first represent the data using projections with largest variance: order and keep the $\omega_i$'s with the largest $\lambda_{max}$.
- In fact, the d x d covariance matrix will not be full rank. Assume you have N training images that are d-dimensional, then you have at most N-1 non-zero eigenvalues.

# Properties of PCA Vectors

- Remember we have computed eigenvectors $\omega$ from covariance matrix $\Sigma$.
- *Important note*: $\Sigma$ matrix is symmetric, i.e., $\Sigma = \Sigma^T$ is how you test for symmetry.
- Proof:

$$\Sigma = E(x - \mu)(x - \mu)^T = [E(x - \mu)(x - \mu)^T]^T$$
$$= E(x - \mu)(x - \mu)^T = \Sigma^T$$

- Symmetric matrix has orthogonal eigenvectors. What does this mean?

$$\omega_i^T \omega_k = 0 \quad \text{for all } i \neq k$$
$$\omega_i^T \omega_k = 1 \quad \text{for all } i = k \quad (\omega^T \omega = 1)$$

# Properties of PCA Vectors

- The dot product between different eigenvectors is 0.
  - They are un-correlated.
  - Information about how data varies in the direction of one of the eigenvectors tells you nothing about how data varies in the directions of the other eigenvectors.

$$\omega_i^T \omega_k = 0 \quad \text{for all } i \neq k$$
$$\omega_i^T \omega_k = 1 \quad \text{for all } i = k \quad (\omega^T \omega = 1)$$

- We have an orthogonal basis that we can use to represent our data.
- Eigenvector $\omega$ = principal direction vector = PCA basis vector

# Representing a Signal Using a Basis

- From this slide on: we refer to principal direction vector ω as **v**
- Now we have a set of N basis vectors $\mathbf{v}_i$ which can be used to represent a signal **x** as follows:

$$\mathbf{x} = \sum_{i=1}^{N} p_i \mathbf{v}_i = p_1 \begin{bmatrix} | \\ \mathbf{v}_1 \\ | \end{bmatrix} + p_2 \begin{bmatrix} | \\ \mathbf{v}_2 \\ | \end{bmatrix} + .. + p_n \begin{bmatrix} | \\ \mathbf{v}_n \\ | \end{bmatrix} = \mathbf{V}\mathbf{p}$$

- i.e., the signal **x** is a linear combination of the basis vectors where the $p_i$ scalars are the weight coefficients.

# Representing a Signal Using a Basis

$$\mathbf{x} = \sum_{i=1}^{N} p_i \mathbf{v}_i = p_1 \begin{bmatrix} | \\ \mathbf{v}_1 \\ | \end{bmatrix} + p_2 \begin{bmatrix} | \\ \mathbf{v}_2 \\ | \end{bmatrix} + .. + p_n \begin{bmatrix} | \\ \mathbf{v}_n \\ | \end{bmatrix} = \mathbf{V}\mathbf{p}$$

- Since this is an orthogonal basis, we can easily find the coefficient *p*.
- These are just the projections of the signal x onto each basis:

$$p_1 = \mathbf{x}^{\mathbf{T}} \mathbf{v}_1 \quad \text{or} \quad \mathbf{p} = \mathbf{V}^{\mathbf{T}} \mathbf{x}$$

$$\mathbf{V}^{\mathbf{T}}\mathbf{x} = \begin{bmatrix} - & \mathbf{v}_1 & - \\ - & \mathbf{v}_2 & - \\ - & \mathbf{v}_3 & - \end{bmatrix} \begin{bmatrix} | \\ \mathbf{x} \\ | \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

# PCA Models Variations about the Mean

- Don't forget that we model the variance about the mean.
- i.e., need to subtract the global mean before we analyze the data:

$$\mathbf{p} = \mathbf{V}^{\mathbf{T}}(\mathbf{x} - \mathbf{m})$$

- By "mean" we mean the sample mean image/vector of all your training data samples.

- Don't forget to add the mean back before you reconstruct the data.

$$\mathbf{x} = \sum_{i=1}^{N} p_i \mathbf{v}_i + \mathbf{m} = p_1 \begin{bmatrix} | \\ \mathbf{v}_1 \\ | \end{bmatrix} + p_2 \begin{bmatrix} | \\ \mathbf{v}_2 \\ | \end{bmatrix} + .. + p_n \begin{bmatrix} | \\ \mathbf{v}_n \\ | \end{bmatrix} + \mathbf{m} = \mathbf{V}\mathbf{p} + \mathbf{m}$$

# PCA Summary

- These projection vectors from a basis for describing the training data in the Minimum Squared Error (MSE) sense.
  - i.e., the eigenvectors with the largest eigenvectors capture most of the signal energy and reconstruction ability.

- N samples of data will have a covariance matrix of rank at most N-1. That means you have at most N-1 non-zero eigenvalues. And you only care about these corresponding N-1 eigenvectors.

- Dimensionality reduction comes when you use N-1 << d eigenvector basis to represent the data (the coefficient vector **p**)
  - In a nutshell, PCA is about learning meaningful (orthogonal) basis from a set of data.

# PCA Basis Visualization

- When the image data is comprised of aligned faces from different people, here are the PCA basis visualized:

# Reconstruction with PCA Basis


1 eig, MSE = 1233.16


2 eigs, MSE = 1027


3 eigs, MSE = 758


4 eigs, MSE = 634


8 eigs, MSE = 285


13 eigs, MSE = 216


20 eigs, MSE = 87


45 eigs, MSE = 6.8

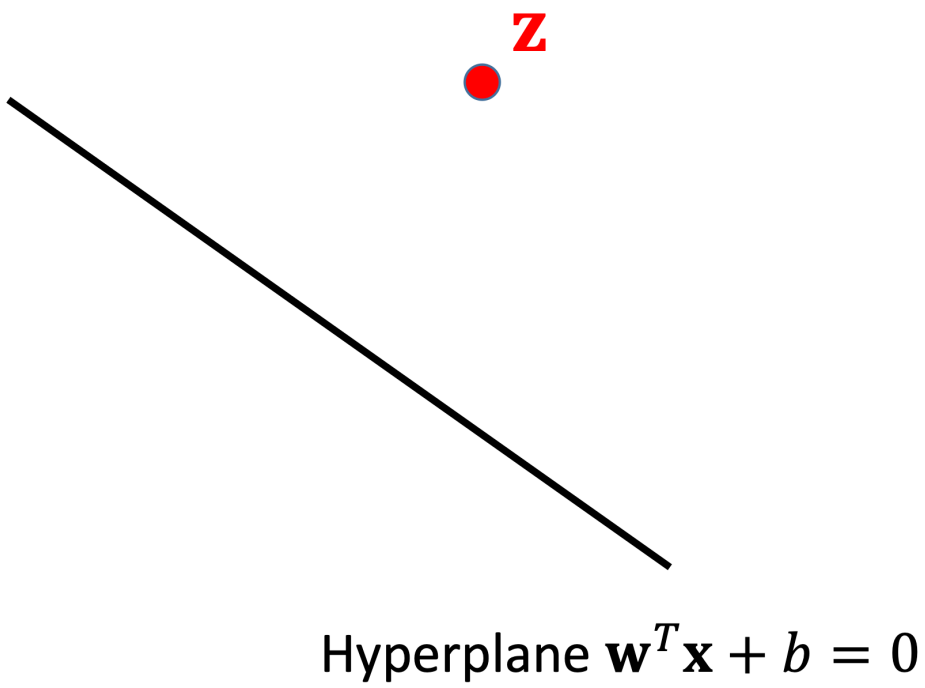

60 eigs, MSE = 0.06


64 eigs, MSE = 0

# Support Vector Machine (SVM)

# – Project a Point onto a Hyperplane

**Question**: how to project $\mathbf{z}$ onto the hyperplane?



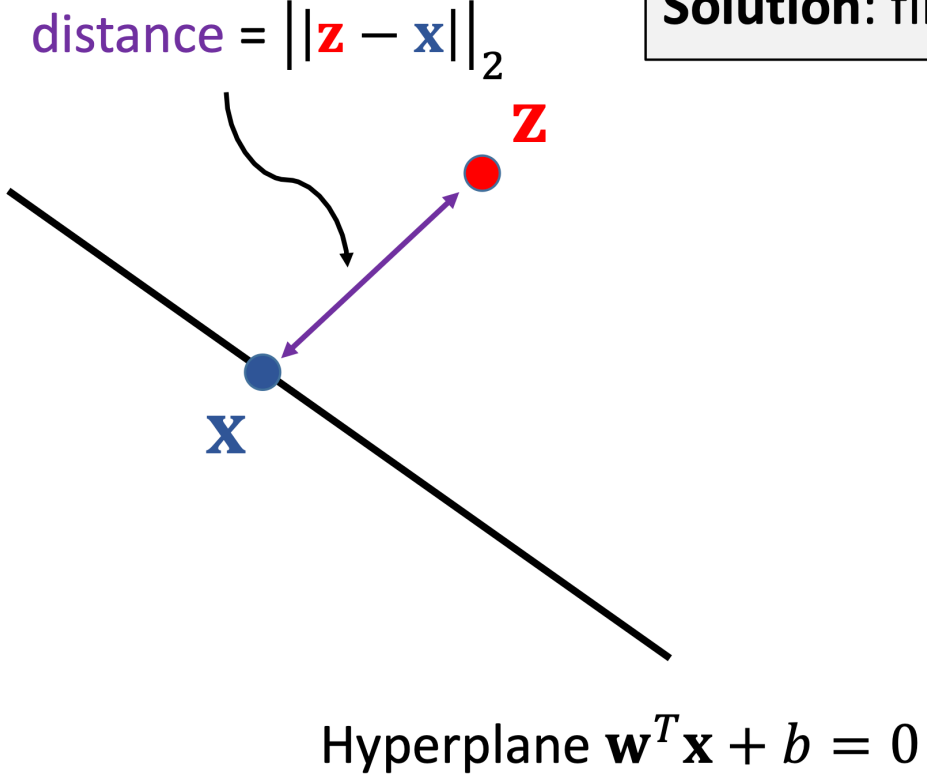

Slides courtesy of Dr. Shusen Wang

# Project a Point onto a Hyperplane

**Question**: how to project $\mathbf{z}$ onto the hyperplane?

**Solution**: find $\mathbf{x}$ on the hyperplane such that $\left|\left|\mathbf{z}-\mathbf{x}\right|\right|_2^2$ is minimized.

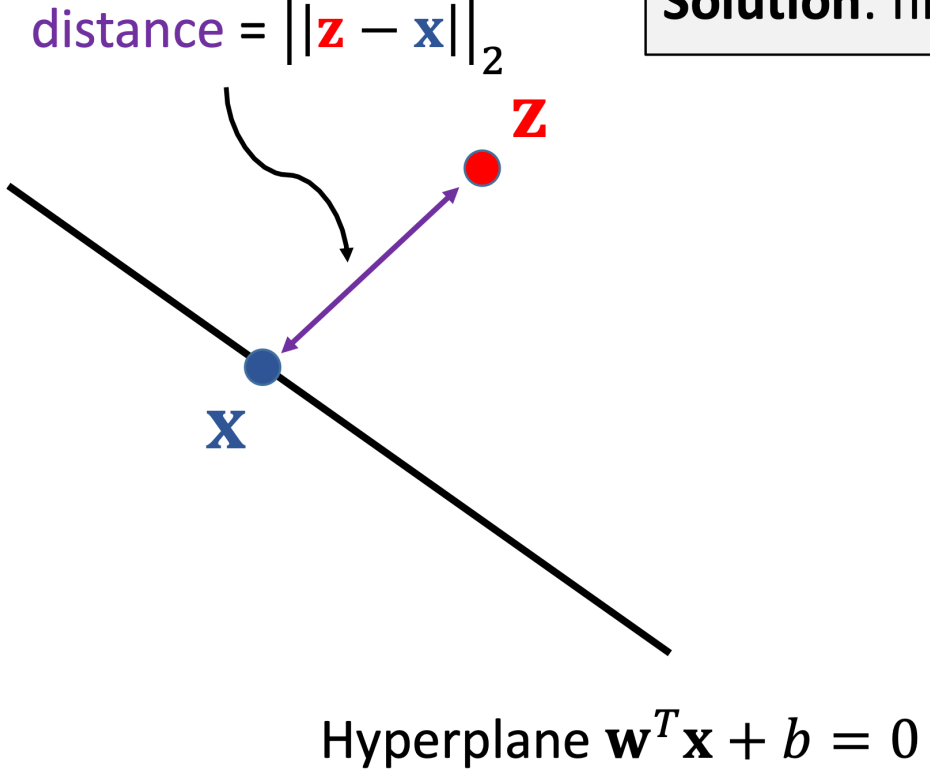
- $\min_{\mathbf{x}} \left|\left|\mathbf{z}-\mathbf{x}\right|\right|_2^2$; s.t. $\mathbf{w}^T\mathbf{x}+b=0$



# Project a Point onto a Hyperplane

**Question**: how to project $\mathbf{z}$ onto the hyperplane?

**Solution**: find $\mathbf{x}$ on the hyperplane such that $\left|\left|\mathbf{z}-\mathbf{x}\right|\right|_2^2$ is minimized.




- $\min_{\mathbf{x}} \left|\left|\mathbf{z}-\mathbf{x}\right|\right|_2^2$; s.t. $\mathbf{w}^T\mathbf{x}+b=0$
- Solve the problem using the KKT conditions:

$$\begin{cases} \dfrac{\partial \left|\left|\mathbf{z}-\mathbf{x}\right|\right|_2^2}{\partial \mathbf{x}} + \lambda \dfrac{\partial\left(\mathbf{w}^T\mathbf{x}+b\right)}{\partial \mathbf{x}} = 0; \\ \mathbf{w}^T\mathbf{x}+b=0. \end{cases}$$

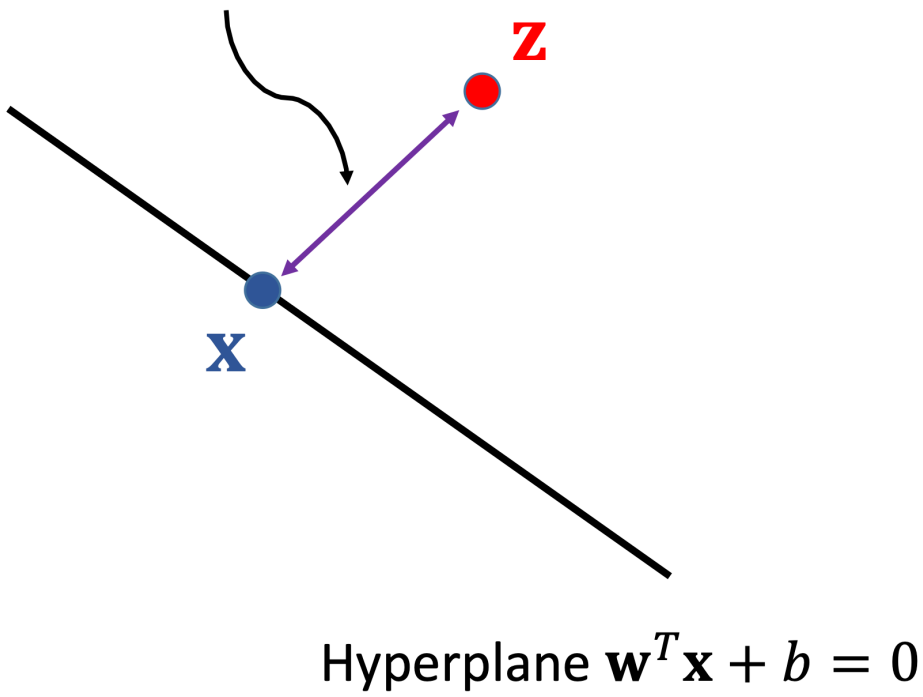
- Solution: $\mathbf{x}=\mathbf{z}-\dfrac{\mathbf{w}^T\mathbf{z}+b}{\left|\left|\mathbf{w}\right|\right|_2^2}\mathbf{w}$

Slides courtesy of Dr. Shusen Wang

# Project a Point onto a Hyperplane

**Question**: how to project $\mathbf{z}$ onto the hyperplane?

**Solution**: find $\mathbf{x}$ on the hyperplane such that $\left|\left|\mathbf{z}-\mathbf{x}\right|\right|_2^2$ is minimized.



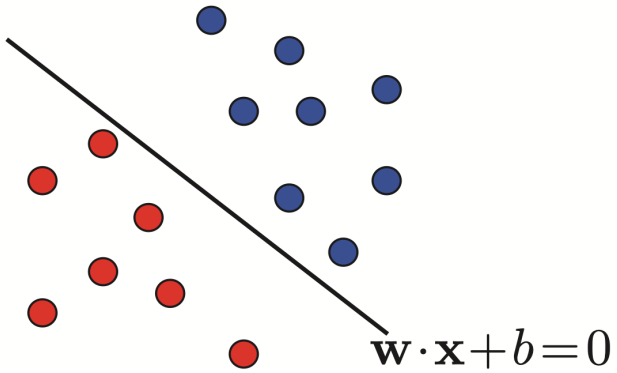

- Solution: $\mathbf{x}=\mathbf{z}-\frac{\mathbf{w}^T\mathbf{z}+b}{\left|\left|\mathbf{w}\right|\right|_2^2}\mathbf{w}$
- The $\ell_2$ distance between $\mathbf{z}$ and the hyperplane is

$$\left|\left|\mathbf{z}-\mathbf{x}\right|\right|_2=\frac{\left|\mathbf{w}^T\mathbf{z}+b\right|}{\left|\left|\mathbf{w}\right|\right|_2}.$$

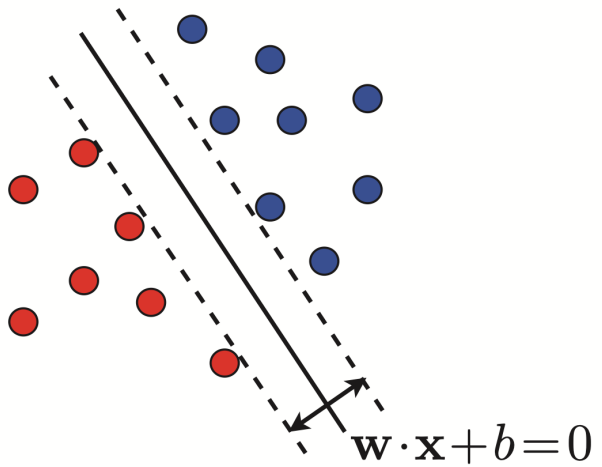
Slides courtesy of Dr. Shusen Wang

# Support Vector Machine (SVM)

Separate data by a hyperplane (assume the data are separable)




An arbitrary hyperplane.



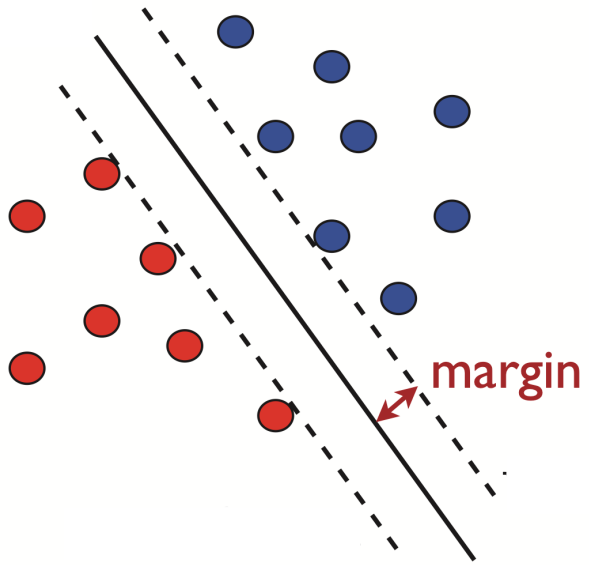

The hyperplane that maximizes the margin.

Slides courtesy of Dr. Shusen Wang

The figure is from the book "*Foundations of Machine Learning*"

# Support Vector Machine (SVM)

Separate data by a hyperplane (assume the data are separable)

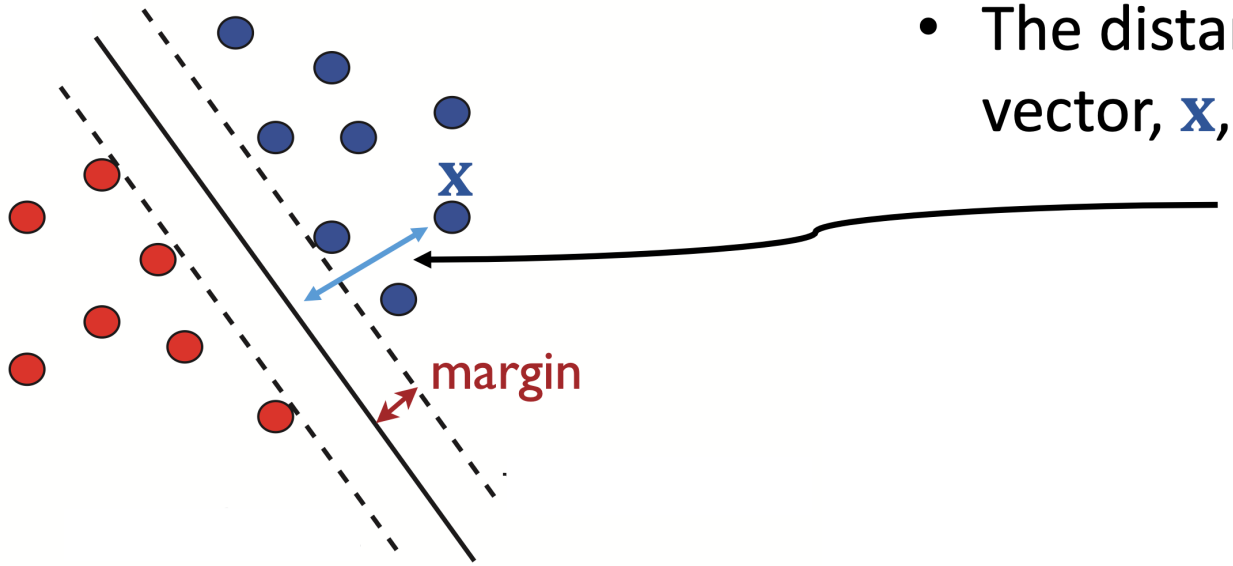



Hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$

The figure is from the book "*Foundations of Machine Learning*"

# Support Vector Machine (SVM)

Separate data by a hyperplane (assume the data are separable)



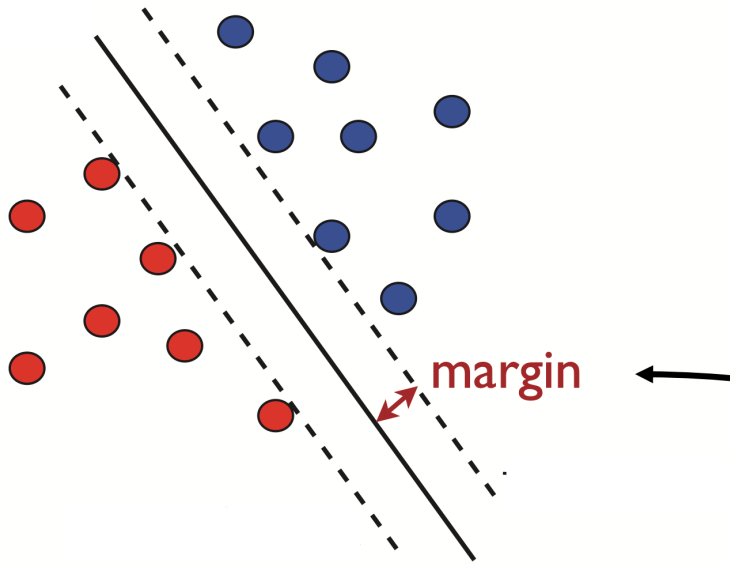

- The distance between any feature vector, $\mathbf{x}$, and the hyperplane is

$$\text{dist} = \frac{|\mathbf{w}^T\mathbf{x}+b|}{||\mathbf{w}||_2}.$$

The figure is from the book "*Foundations of Machine Learning*"

# – Support Vector Machine (SVM)

Separate data by a hyperplane (assume the data are separable)




Hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$

- The distance between any feature vector, $\mathbf{x}$, and the hyperplane is

$$\text{dist} = \frac{|\mathbf{w}^T\mathbf{x}+b|}{||\mathbf{w}||_2}.$$

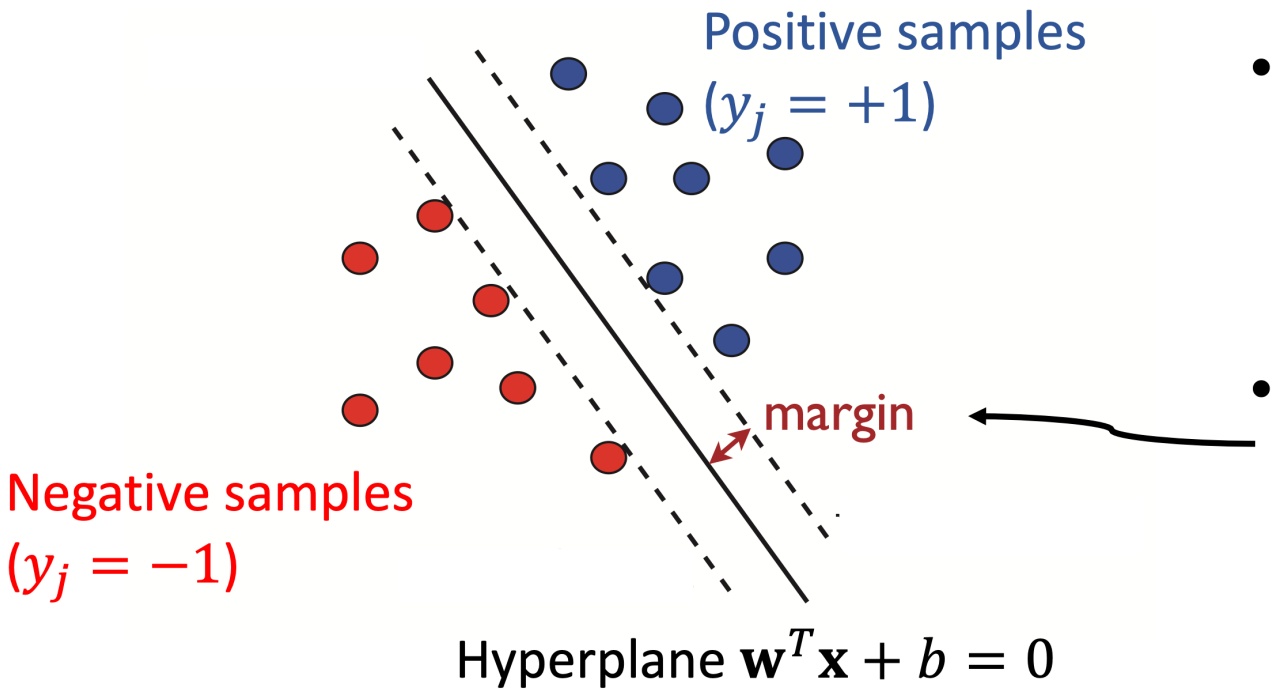
- The margin is the smallest distance:

$$\min_j \frac{|\mathbf{w}^T\mathbf{x}_j+b|}{||\mathbf{w}||_2}$$

The figure is from the book "*Foundations of Machine Learning*"

# – Support Vector Machine (SVM)

Separate data by a hyperplane (assume the data are separable)




- The distance between any feature vector, $\mathbf{x}$, and the hyperplane is

$$\text{dist} = \frac{|\mathbf{w}^T\mathbf{x}+b|}{||\mathbf{w}||_2}.$$

- The margin is the smallest distance:

$$\min_j \frac{|\mathbf{w}^T\mathbf{x}_j+b|}{||\mathbf{w}||_2} = \min_j \frac{y_j(\mathbf{w}^T\mathbf{x}_j+b)}{||\mathbf{w}||_2}$$

The figure is from the book "*Foundations of Machine Learning*"

# – Support Vector Machine (SVM)

Margin = $\min_j \frac{y_j(\mathbf{w}^T\mathbf{x}_j+b)}{||\mathbf{w}||_2}$; we want to maximize the margin.

Slides courtesy of Dr. Shusen Wang

# Support Vector Machine (SVM)

Margin = $\min_j \frac{y_j(\mathbf{w}^T\mathbf{x}_j+b)}{||\mathbf{w}||_2}$; we want to maximize the margin.

Define $\bar{\mathbf{x}}_j = [\mathbf{x}_j; 1] \in \mathbb{R}^{d+1}$

Define $\bar{\mathbf{w}} = [\mathbf{w}, b] \in \mathbb{R}^{d+1}$

➔ $\mathbf{x}_j^T\mathbf{w} + b = \bar{\mathbf{x}}_j^T\bar{\mathbf{w}}$

# Support Vector Machine (SVM)

Margin = $\min_j \frac{y_j \mathbf{w}^T \mathbf{x}_j}{||\mathbf{w}||_2}$; we want to maximize the margin.

Support Vector Machine (SVM): $\max_{\mathbf{w}} \min_j \frac{y_j \mathbf{w}^T \mathbf{x}_j}{||\mathbf{w}||_2}$

Slides courtesy of Dr. Shusen Wang

# – Support Vector Machine (SVM)

Support Vector Machine (SVM): $\max_{\mathbf{w}} \min_{j} \frac{y_j \mathbf{w}^T \mathbf{x}_j}{||\mathbf{w}||_2}$

Slides courtesy of Dr. Shusen Wang

# Support Vector Machine (SVM)

Support Vector Machine (SVM): $\max_{\mathbf{w}} \min_{j} \frac{y_j \mathbf{w}^T \mathbf{x}_j}{||\mathbf{w}||_2}$

$$\underset{\mathbf{w}}{\text{argmax}} \min_{j} \frac{y_j \mathbf{w}^T \mathbf{x}_j}{||\mathbf{w}||_2} = \underset{\mathbf{w}}{\text{argmax}} \frac{\min_{j} y_j \mathbf{w}^T \mathbf{x}_j}{||\mathbf{w}||_2}$$

$$= \underset{\mathbf{w}}{\text{argmax}} \frac{1}{||\mathbf{w}||_2}, \quad \text{s.t.} \quad \left(\min_{j} y_j \mathbf{w}^T \mathbf{x}_j\right) = 1$$

$$= \underset{\mathbf{w}}{\text{argmin}} ||\mathbf{w}||_2^2, \quad \text{s.t.} \quad \left(\min_{j} y_j \mathbf{w}^T \mathbf{x}_j\right) = 1$$

$$= \underset{\mathbf{w}}{\text{argmin}} ||\mathbf{w}||_2^2, \quad \text{s.t.} \quad y_j \mathbf{w}^T \mathbf{x}_j \geq 1 \text{ for all } j$$

Slides courtesy of Dr. Shusen Wang

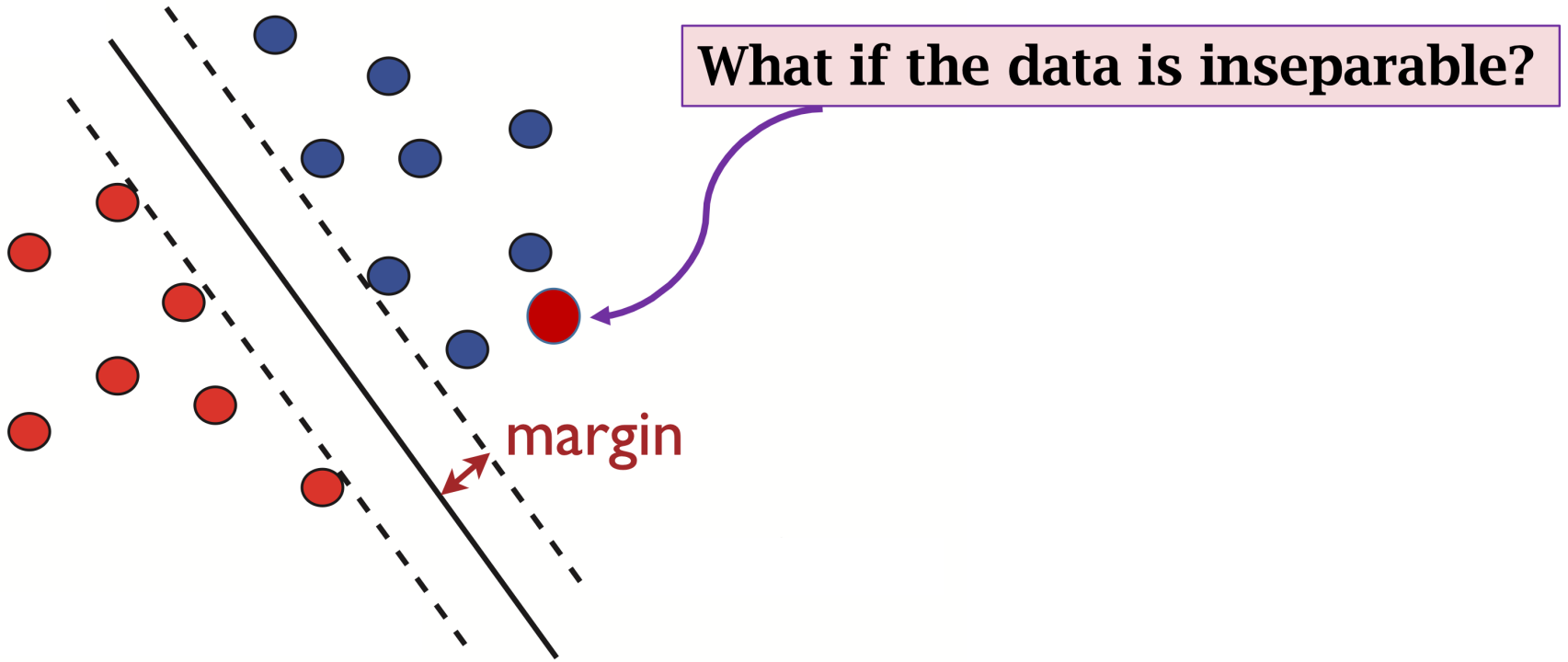
# Support Vector Machine (SVM)

$$\min_{\mathbf{w}} ||\mathbf{w}||_2^2, \quad \text{s.t.} \quad y_j \mathbf{w}^T \mathbf{x}_j \geq 1 \text{ for all } j \in \{1, \cdots, n\}.$$

**Equivalent form of SVM**

# Support Vector Machine (SVM)

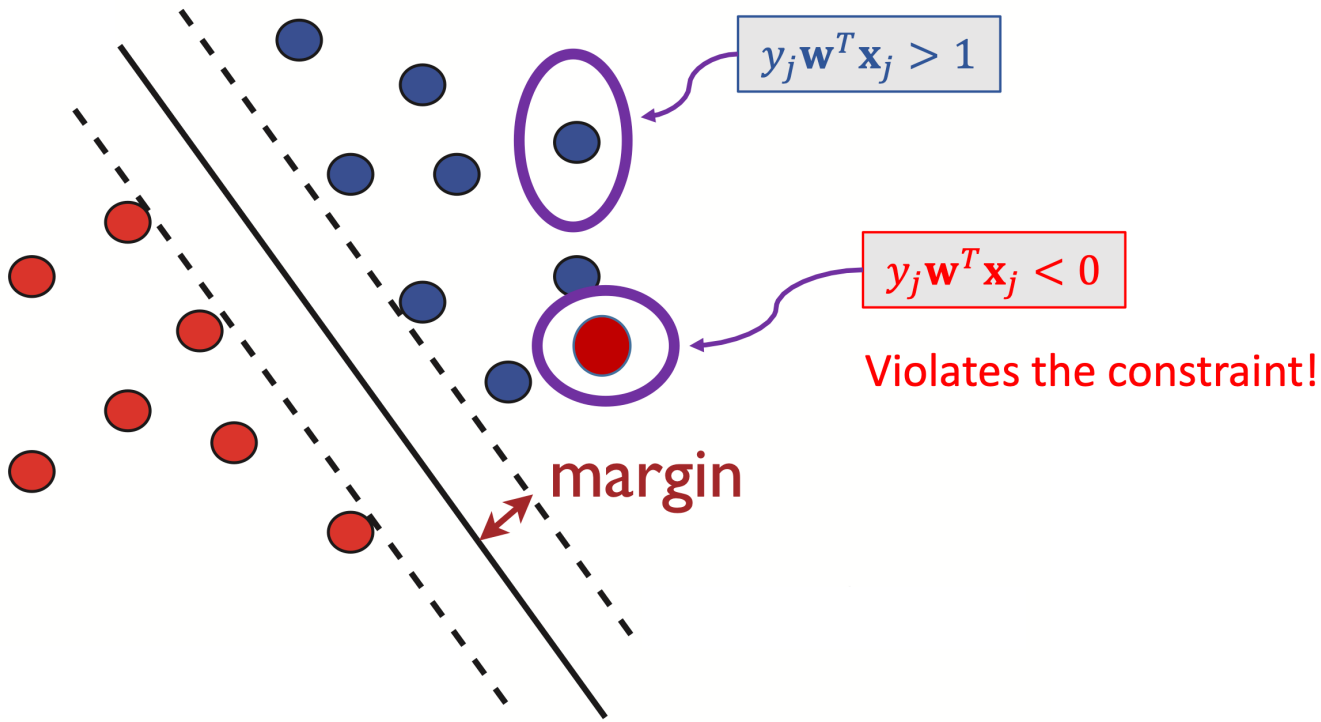
$$\min_{\mathbf{w}} ||\mathbf{w}||_2^2, \quad \text{s.t.} \quad y_j \mathbf{w}^T \mathbf{x}_j \geq 1 \text{ for all } j \in \{1, \cdots, n\}.$$




Slides courtesy of Dr. Shusen Wang

# Support Vector Machine (SVM)

$$\min_{\mathbf{w}} ||\mathbf{w}||_2^2, \quad \text{s.t.} \quad y_j \mathbf{w}^T \mathbf{x}_j \geq 1 \text{ for all } j \in \{1, \cdots, n\}.$$




Slides courtesy of Dr. Shusen Wang

# Support Vector Machine (SVM)

$$\min_{\mathbf{w}} \left|\left|\mathbf{w}\right|\right|_2^2, \quad \text{s.t.} \quad 1 - y_j \mathbf{w}^T \mathbf{x}_j \leq 0 \text{ for all } j \in \{1, \cdots, n\}.$$

Relax

$$\min_{\mathbf{w}, \xi_j} \left|\left|\mathbf{w}\right|\right|_2^2 + \lambda \sum_j [\xi_j]_+, \quad \text{s.t.} \quad 1 - y_j \mathbf{w}^T \mathbf{x}_j = \xi_j \text{ for all } j \in \{1, \cdots, n\}.$$

- $[\xi_j]_+ = \max\{\xi_j, 0\}$

# Support Vector Machine (SVM)

$$\min_{\mathbf{w}} \left|\left|\mathbf{w}\right|\right|_2^2, \quad \text{s.t.} \quad 1 - y_j \mathbf{w}^T \mathbf{x}_j \leq 0 \text{ for all } j \in \{1, \cdots, n\}.$$

Relax

$$\min_{\mathbf{w}, \xi_j} \left|\left|\mathbf{w}\right|\right|_2^2 + \lambda \sum_j [\xi_j]_+, \quad \text{s.t.} \quad 1 - y_j \mathbf{w}^T \mathbf{x}_j = \xi_j \text{ for all } j \in \{1, \cdots, n\}.$$

- $[\xi_j]_+ = \max\{\xi_j, 0\}$
- $\xi_j \leq 0$ means the constraint $1 - y_j \mathbf{w}^T \mathbf{x}_j \leq 0$ is satisfied
  ➔ no penalty!
- $\xi_j > 0$ means the constraint is violated (because the data is inseparable)
  ➔ penalize the violation $\xi_j$.

Slides courtesy of Dr. Shusen Wang

# Support Vector Machine (SVM)

$$\min_{\mathbf{w}} \left|\left|\mathbf{w}\right|\right|_2^2, \quad \text{s.t.} \quad 1 - y_j \mathbf{w}^T \mathbf{x}_j \leq 0 \text{ for all } j \in \{1, \cdots, n\}.$$

Relax

$$\min_{\mathbf{w}, \xi_j} \left|\left|\mathbf{w}\right|\right|_2^2 + \lambda \sum_j [\xi_j]_+, \quad \text{s.t.} \quad 1 - y_j \mathbf{w}^T \mathbf{x}_j = \xi_j \text{ for all } j \in \{1, \cdots, n\}.$$

Equivalent

$$\min_{\mathbf{w}, b} \left|\left|\mathbf{w}\right|\right|_2^2 + \lambda \sum_j [1 - y_j \mathbf{w}^T \mathbf{x}_j]_+.$$

Slides courtesy of Dr. Shusen Wang

# Clustering with K-Means

# Clustering Task

**Tasks**

clustering

**Methods**

K-means

**Algorithms**

Lloyd's algorithm

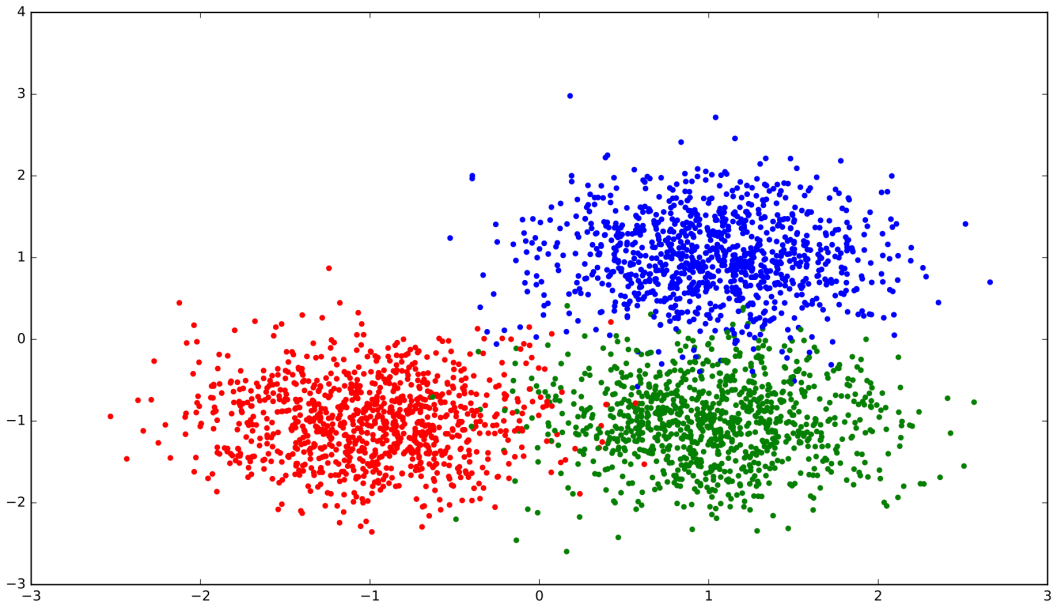
Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Clustering Task

**Input:** vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and cluster number $k$ ($\ll n$).

**Output:** predicted class labels $y_1, \cdots, y_n \in \{1, 2, \cdots, k\}$.



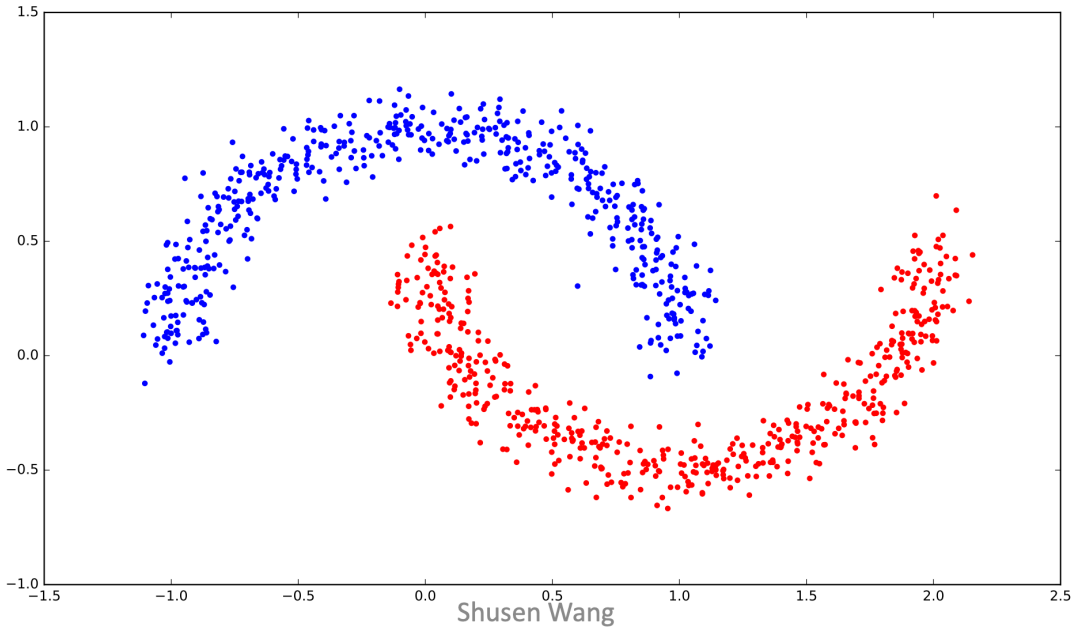

$k = 3$

Slides courtesy of Dr. Shusen Wang

# Clustering Task

**Input:** vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and cluster number $k$ ($\ll n$).

**Output:** predicted class labels $y_1, \cdots, y_n \in \{1, 2, \cdots, k\}$.



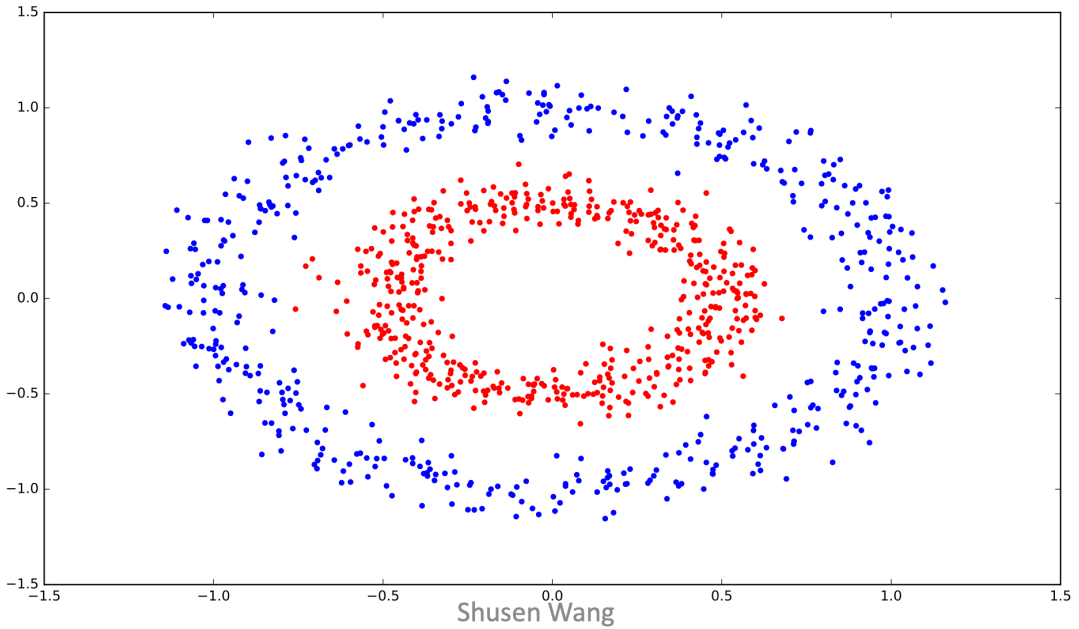

$k = 2$

Slides courtesy of Dr. Shusen Wang

# Clustering Task

**Input:** vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and cluster number $k$ ($\ll n$).

**Output:** predicted class labels $y_1, \cdots, y_n \in \{1, 2, \cdots, k\}$.




$k = 2$

Slides courtesy of Dr. Shusen Wang

# Clustering Task

**Input:** vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and cluster number $k$ ($\ll n$).

**Output:** predicted class labels $y_1, \cdots, y_n \in [k]$.

denote $[k] = \{1, 2, \cdots, k\}$

# Clustering Task

**Input:** vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and cluster number $k$ ($\ll n$).

**Output:** predicted class labels $y_1, \cdots, y_n \in [k]$.

**Equivalent definition:**

**Clustering:** partition $[n]$ to $k$ disjoint subsets $S_1, \cdots, S_k$,

- $S_1 \cup \cdots \cup S_k = [n]$,
- $S_i \cap S_j = \emptyset$, for all $i \neq j$.

Shusen Wang

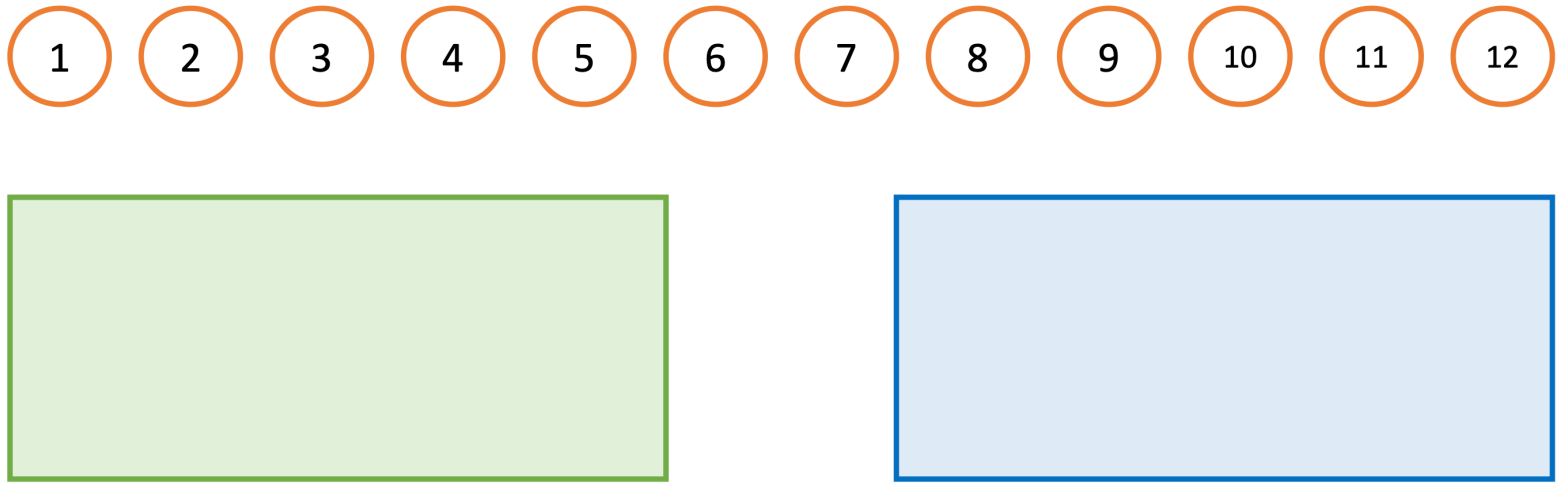
# Clustering Task

**Clustering:** partition $[n]$ to $k$ disjoint subsets $S_1, \cdots, S_k$,

- $S_1 \cup \cdots \cup S_k = [n]$,
- $S_i \cap S_j = \emptyset$, for all $i \neq j$.

Example

- $n = 12$
- $k = 2$




Shusen Wang

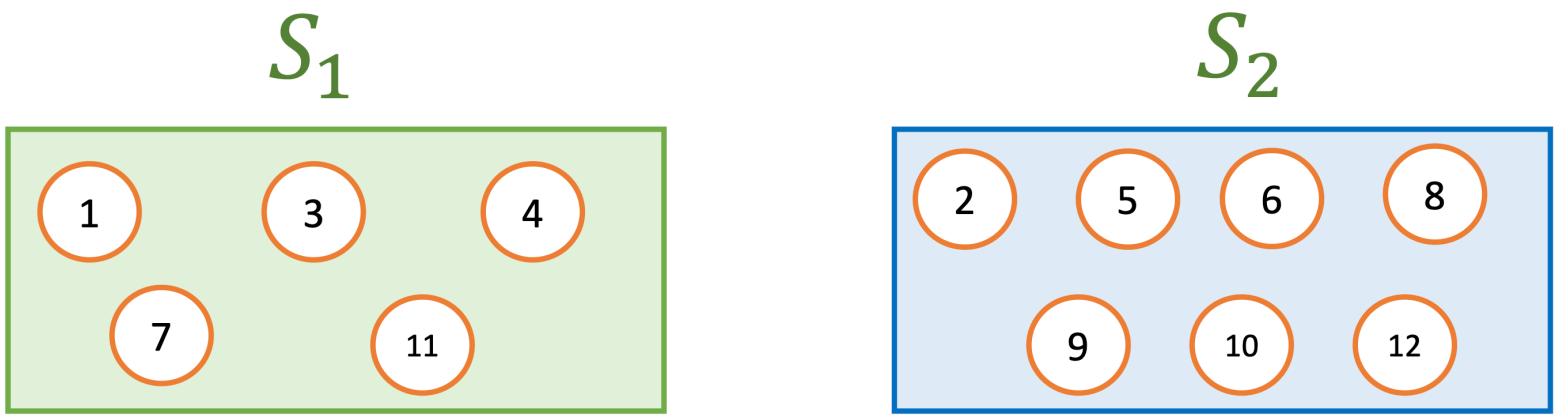
# Clustering Task

**Clustering:** partition $[n]$ to $k$ disjoint subsets $S_1, \cdots, S_k$,

- $S_1 \cup \cdots \cup S_k = [n]$,
- $S_i \cap S_j = \emptyset$, for all $i \neq j$.

Example

- $n = 12$
- $k = 2$




Shusen Wang

# K-Means Clustering Method

Tasks

clustering

**Methods**

K-means

Algorithms

Lloyd's algorithm

Shusen Wang

Slides courtesy of Dr. Shusen Wang

# K-Means Clustering Method

**Clustering:** partition $[n]$ to $k$ disjoint subsets $S_1, \cdots, S_k$,

- $S_1 \cup \cdots \cup S_k = [n]$,
- $S_i \cap S_j = \emptyset$, for all $i \neq j$.

**K-Means Model:**

$$\min_{S_1,\cdots,S_k} \sum_{i=1}^{k} \sum_{j \in S_i} \left\| \mathbf{x}_j - \frac{1}{|S_i|} \sum_{l \in S_i} \mathbf{x}_l \right\|_2^2 .$$

- $\mathbf{x}_j$ indexed by $S_i$

Shusen Wang

Slides courtesy of Dr. Shusen Wang

# K-Means Clustering Method

**Clustering:** partition $[n]$ to $k$ disjoint subsets $S_1, \cdots, S_k$,

- $S_1 \cup \cdots \cup S_k = [n]$,
- $S_i \cap S_j = \emptyset$, for all $i \neq j$.

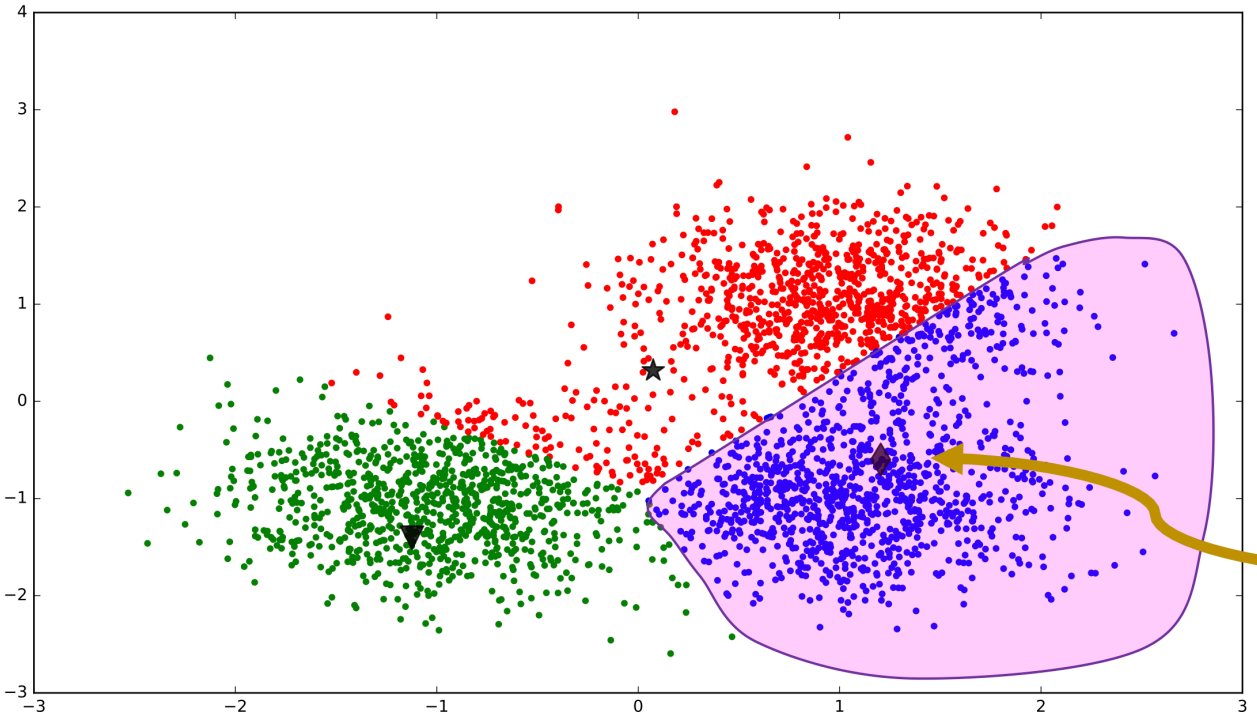
**K-Means Model:**

$$\min_{S_1,\cdots,S_k} \sum_{i=1}^{k} \sum_{j \in S_i} \left\| \mathbf{x}_j - \frac{1}{|S_i|} \sum_{l \in S_i} \mathbf{x}_l \right\|_2^2 .$$

- $\mathbf{x}_j$ indexed by $S_i$
- The centroid of the $i$-th cluster.

Shusen Wang

# K-Means Clustering Method




- $\mathbf{x}_j$ indexed by $S_i$

- The centroid of the $i$-th cluster.

Shusen Wang

Slides courtesy of Dr. Shusen Wang

# K-Means Clustering Method

**Clustering:** partition $[n]$ to $k$ disjoint subsets $S_1, \cdots, S_k$,

- $S_1 \cup \cdots \cup S_k = [n]$,
- $S_i \cap S_j = \emptyset$, for all $i \neq j$.

**K-Means Model:**

$$\min_{S_1,\cdots,S_k} \sum_{i=1}^{k} \sum_{j \in S_i} \left\| \left\| \mathbf{x}_j - \frac{1}{|S_i|} \sum_{l \in S_i} \mathbf{x}_l \right\|_2 \right\|_2^2$$

- Squared Euclean distance between $\mathbf{x}_j$ and its cluster centroid.

Shusen Wang

# K-Means Clustering Method

**Clustering:** partition $[n]$ to $k$ disjoint subsets $S_1, \cdots, S_k$,

- $S_1 \cup \cdots \cup S_k = [n]$,
- $S_i \cap S_j = \emptyset$, for all $i \neq j$.

**K-Means Model:**

$$\min_{S_1,\cdots,S_k} \sum_{i=1}^{k} \sum_{j \in S_i} \left\| \mathbf{x}_j - \frac{1}{|S_i|} \sum_{l \in S_i} \mathbf{x}_l \right\|_2^2 .$$

- It is a combinatorial optimization problem.
- NP-hard!

Shusen Wang

# Lloyd's Algorithm

| Tasks | Methods | Algorithms |
| --- | --- | --- |
| clustering | K-means | Lloyd's algorithm |

Shusen Wang

# Lloyd's Algorithm

**Input:** vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and cluster number $k$ ($\ll n$).

1. Initialize cluster centroids $\mathbf{c}_1, \cdots, \mathbf{c}_k \in \mathbb{R}^d$.

**Example 1:** Randomly select $k$ points from $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ as the cluster centroids .

Shusen Wang

# Lloyd's Algorithm

**Input:** vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and cluster number $k$ ($\ll n$).

1. Initialize cluster centroids $\mathbf{c}_1, \cdots, \mathbf{c}_k \in \mathbb{R}^d$.

**Example 2:**

- Randomly select one point from $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ as the 1st centroid, $\mathbf{c}_1$.
- Select the point from $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ as $\mathbf{c}_2$ by maximizing $\left|\left|\mathbf{x}_j - \mathbf{c}_1\right|\right|_2^2$.
- Select the point from $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ as $\mathbf{c}_3$ by maximizing $\left|\left|\mathbf{x}_j - \mathbf{c}_1\right|\right|_2^2 + \left|\left|\mathbf{x}_j - \mathbf{c}_2\right|\right|_2^2$.
- And so on…

# Lloyd's Algorithm

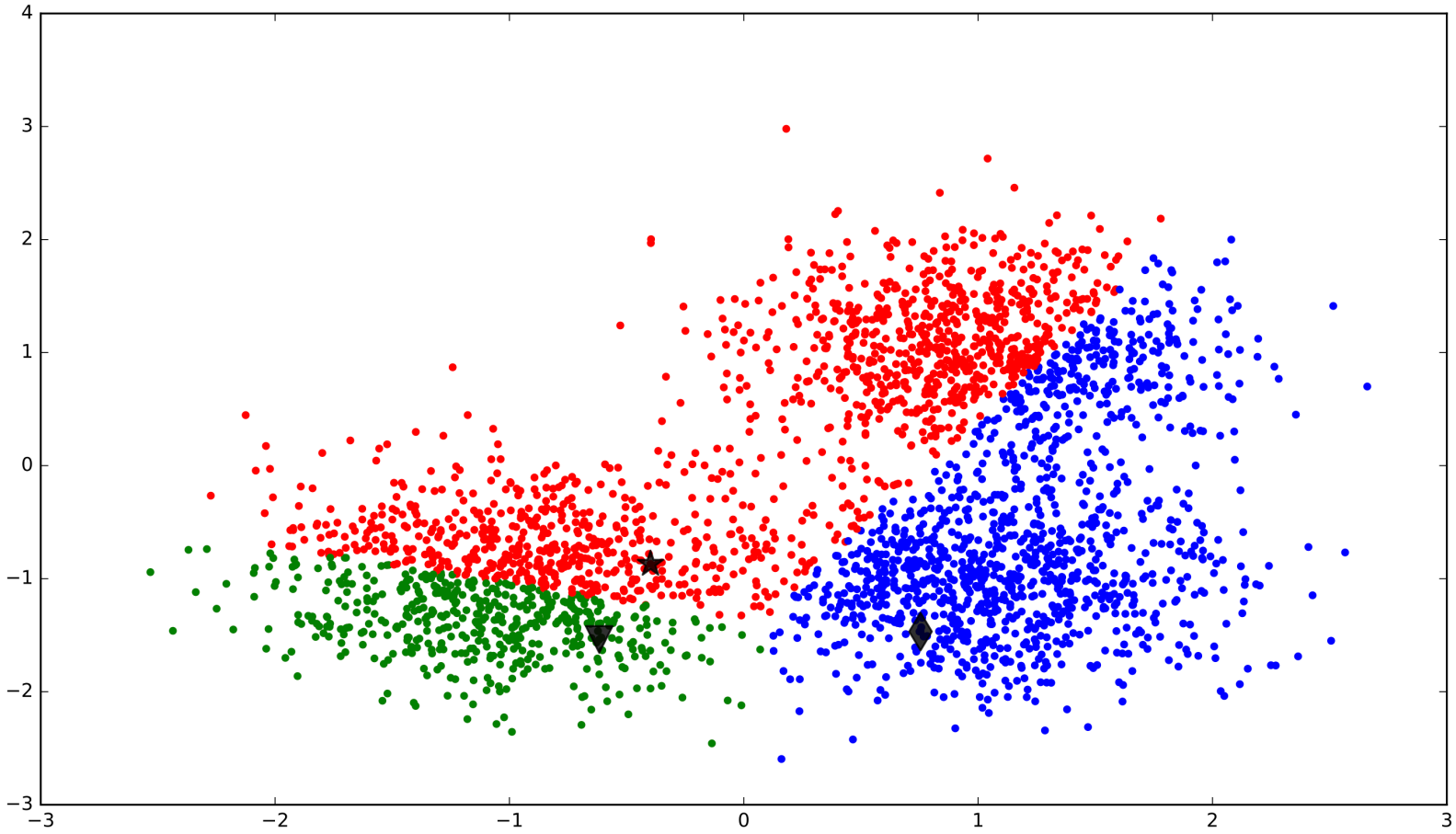
**Input:** vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and cluster number $k$ ($\ll n$).

1. Initialize cluster centroids $\mathbf{c}_1, \cdots, \mathbf{c}_k \in \mathbb{R}^d$.
2. Repeat
   i. For $j = 1$ to $n$, move $j$ to set $S_i$, where $i$ optimizes

$$\min_{i \in [k]} || \mathbf{x}_j - c_i | \Big|_2^2 .$$

Assign $\mathbf{x}_j$ to its nearest centroid.

# Lloyd's Algorithm




Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Lloyd's Algorithm

**Input:** vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and cluster number $k$ ($\ll n$).

1. Initialize cluster centroids $\mathbf{c}_1, \cdots, \mathbf{c}_k \in \mathbb{R}^d$.
2. Repeat
   i. For $j = 1$ to $n$, move $j$ to set $S_i$, where $i$ optimizes
   $$\min_{i \in [k]} || \mathbf{x}_j - c_i | \Big|_2^2 .$$
   ii. For $i = 1$ to $k$, re-compute centroids by $\tilde{\mathbf{c}}_i = \frac{1}{|S_i|} \sum_{j \in S_i} \mathbf{x}_j$ .
   - New centroid $\mathbf{c}_i$: the feature vector (among $\mathbf{x}_1, \cdots, \mathbf{x}_n$) closest to $\tilde{\mathbf{c}}_i$.

move centroids

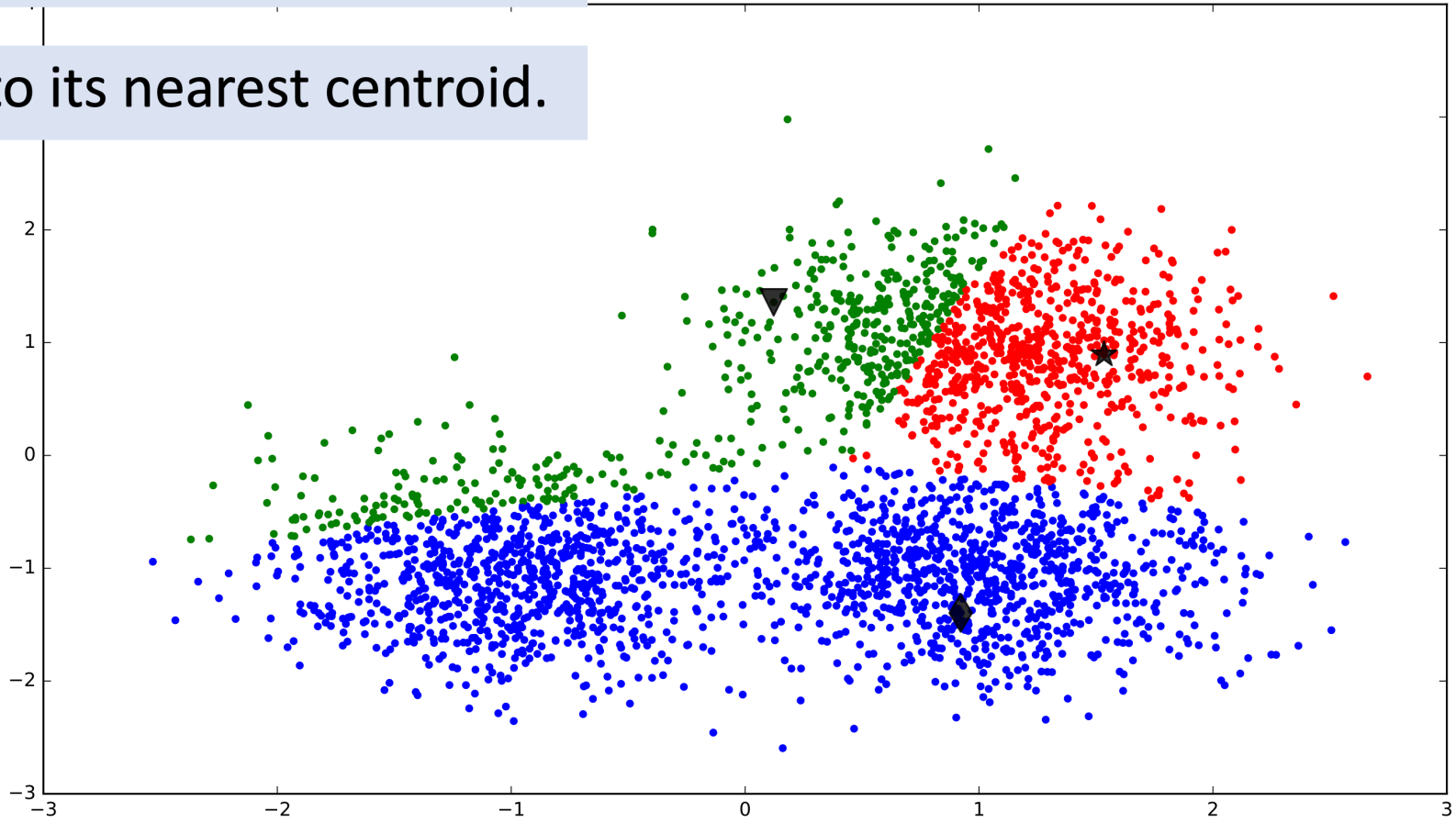
Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Lloyd's Algorithm

Random initialization.

Assign $\mathbf{x}_j$ to its nearest centroid.

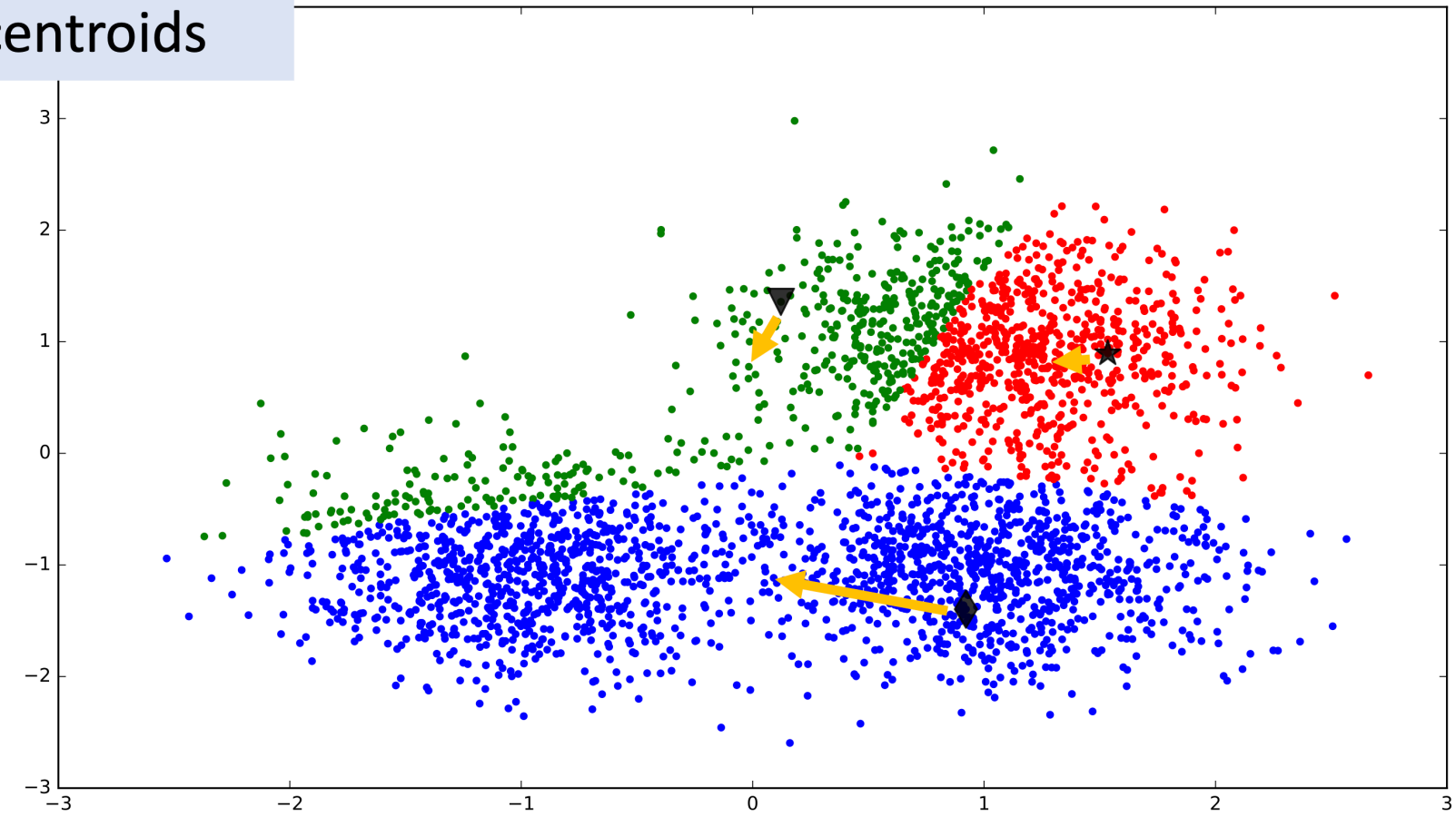



Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Lloyd's Algorithm

move centroids

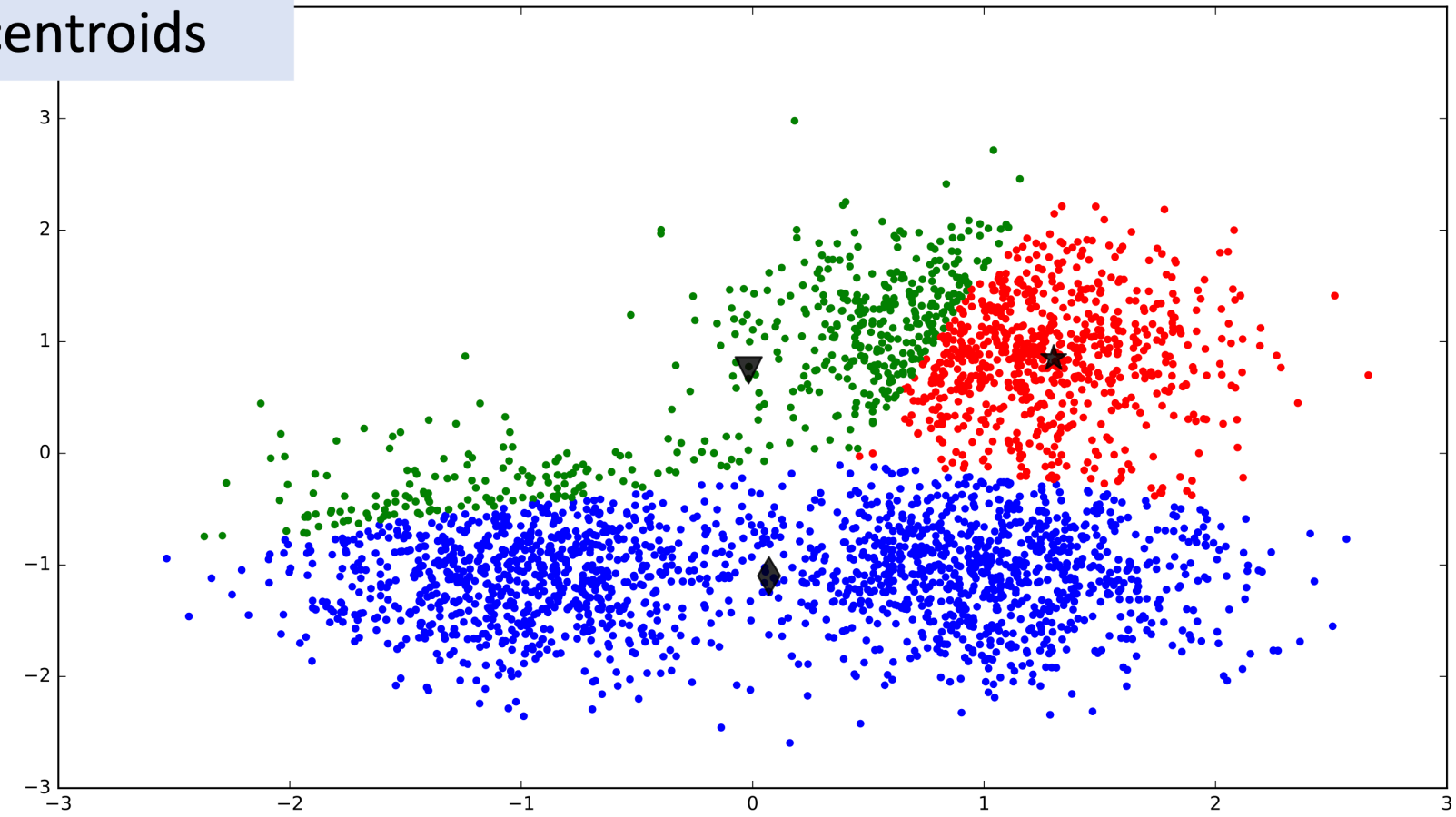



Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Lloyd's Algorithm

move centroids

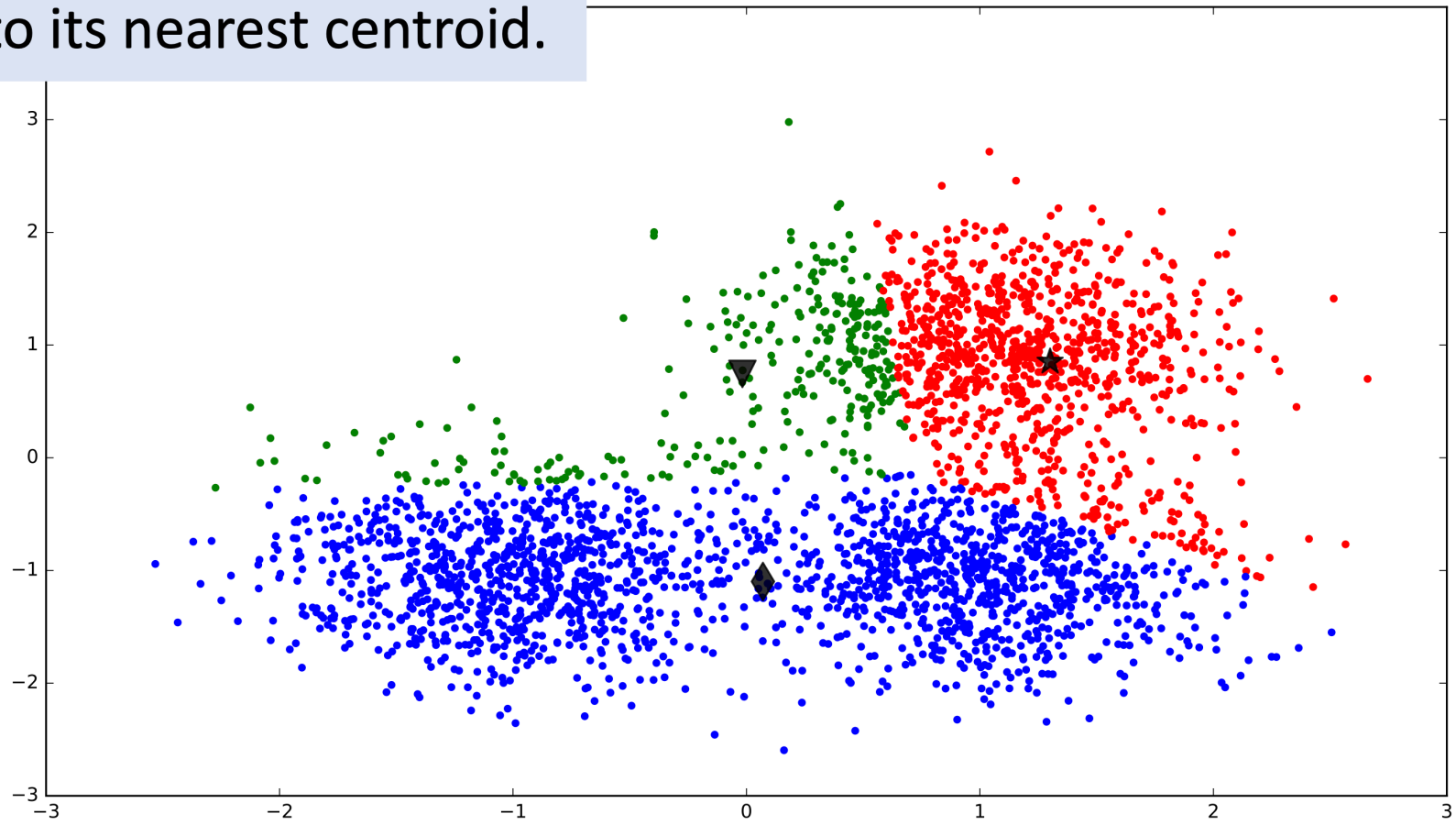



Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Lloyd's Algorithm

Assign $\mathbf{x}_j$ to its nearest centroid.

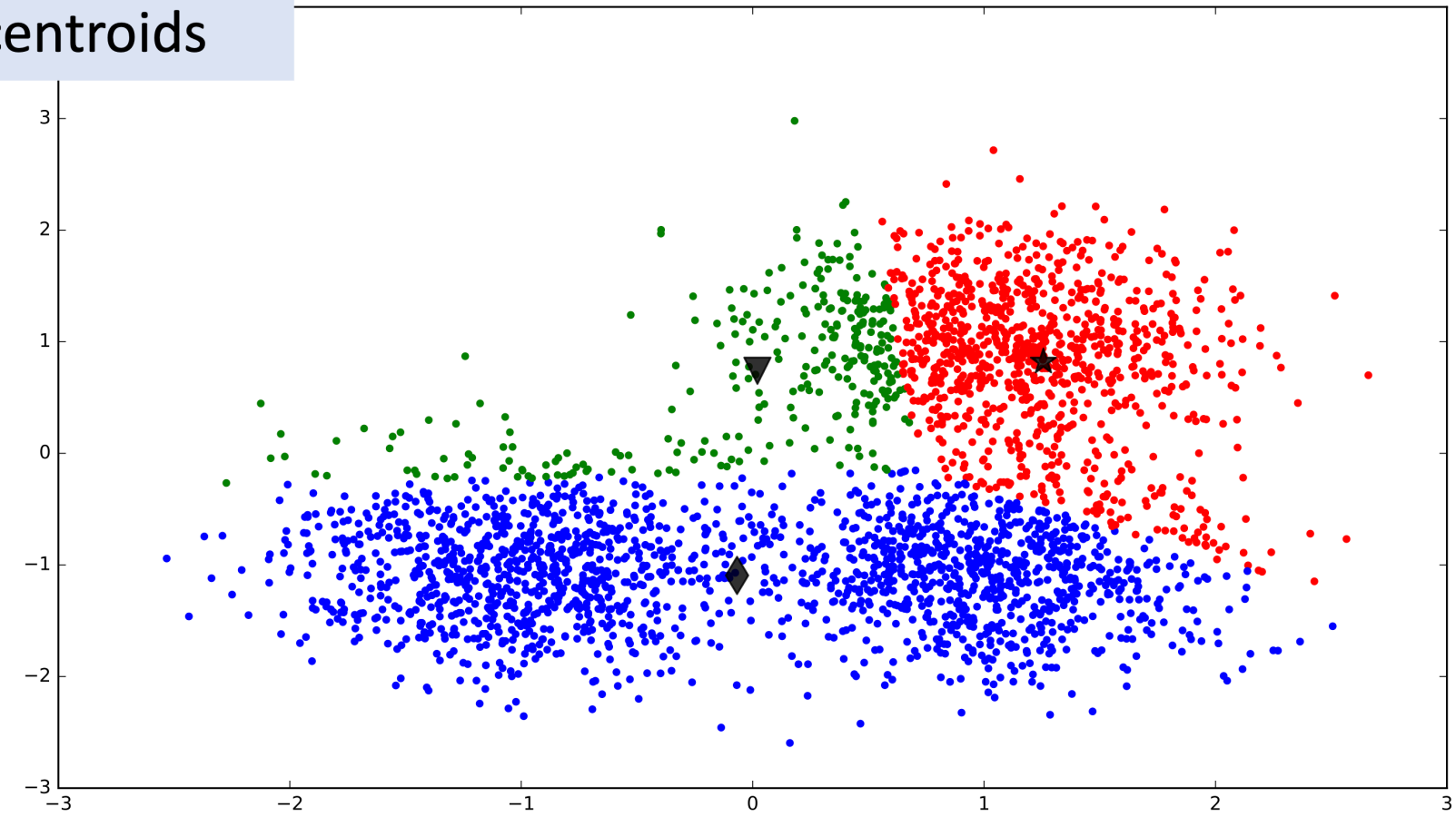



Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Lloyd's Algorithm

move centroids

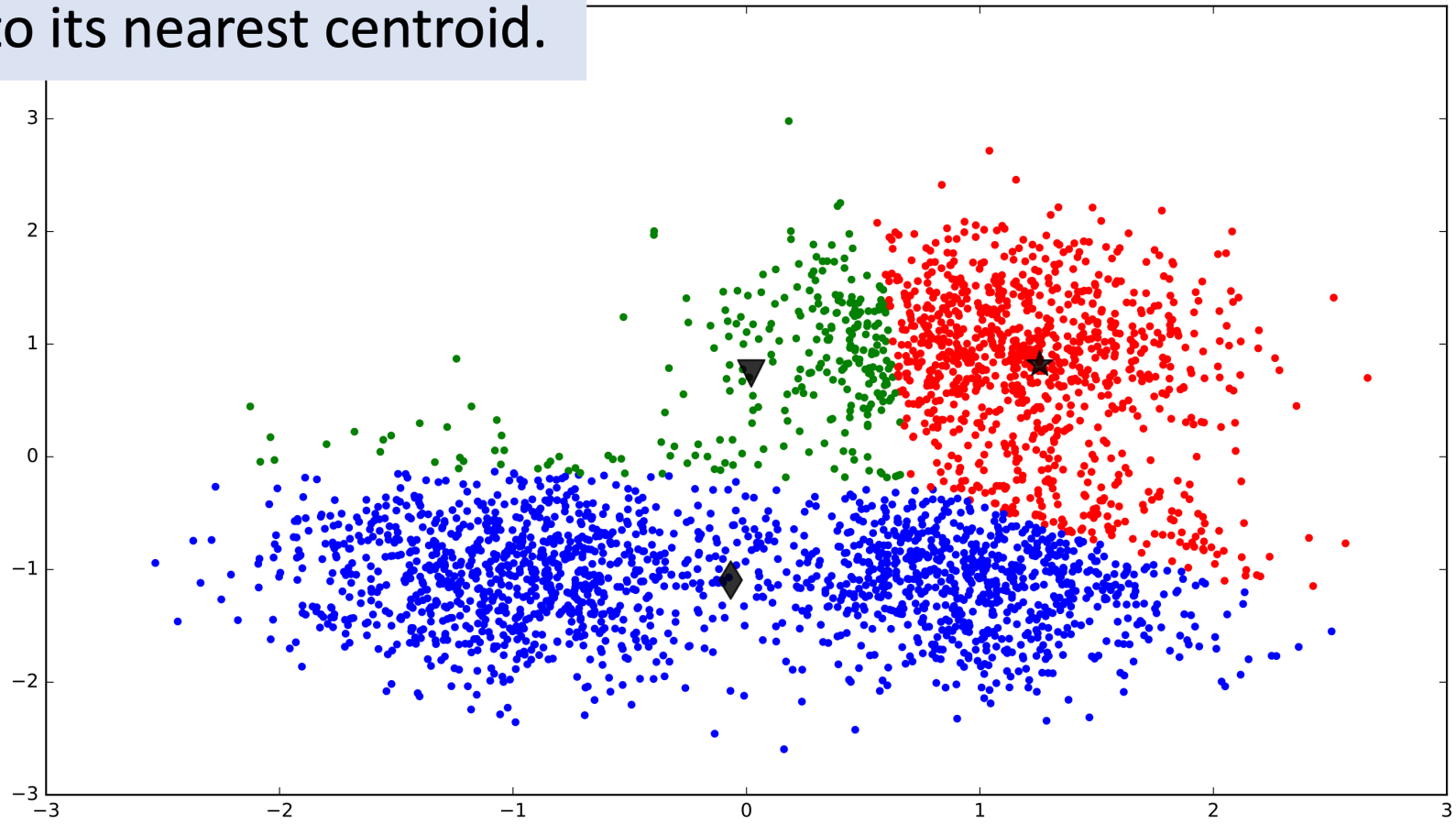



Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Lloyd's Algorithm

Assign $\mathbf{x}_j$ to its nearest centroid.

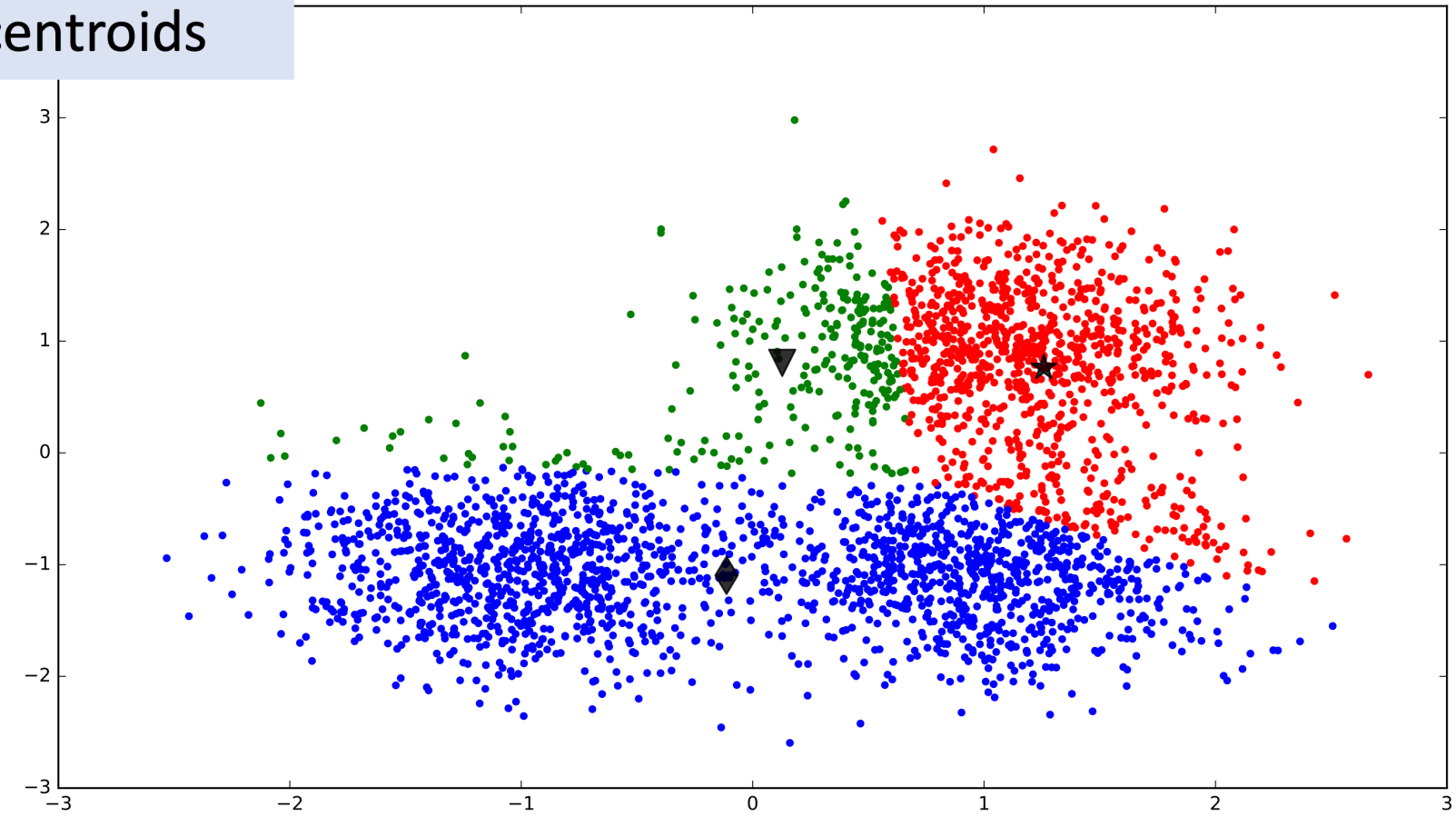



Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Lloyd's Algorithm

move centroids

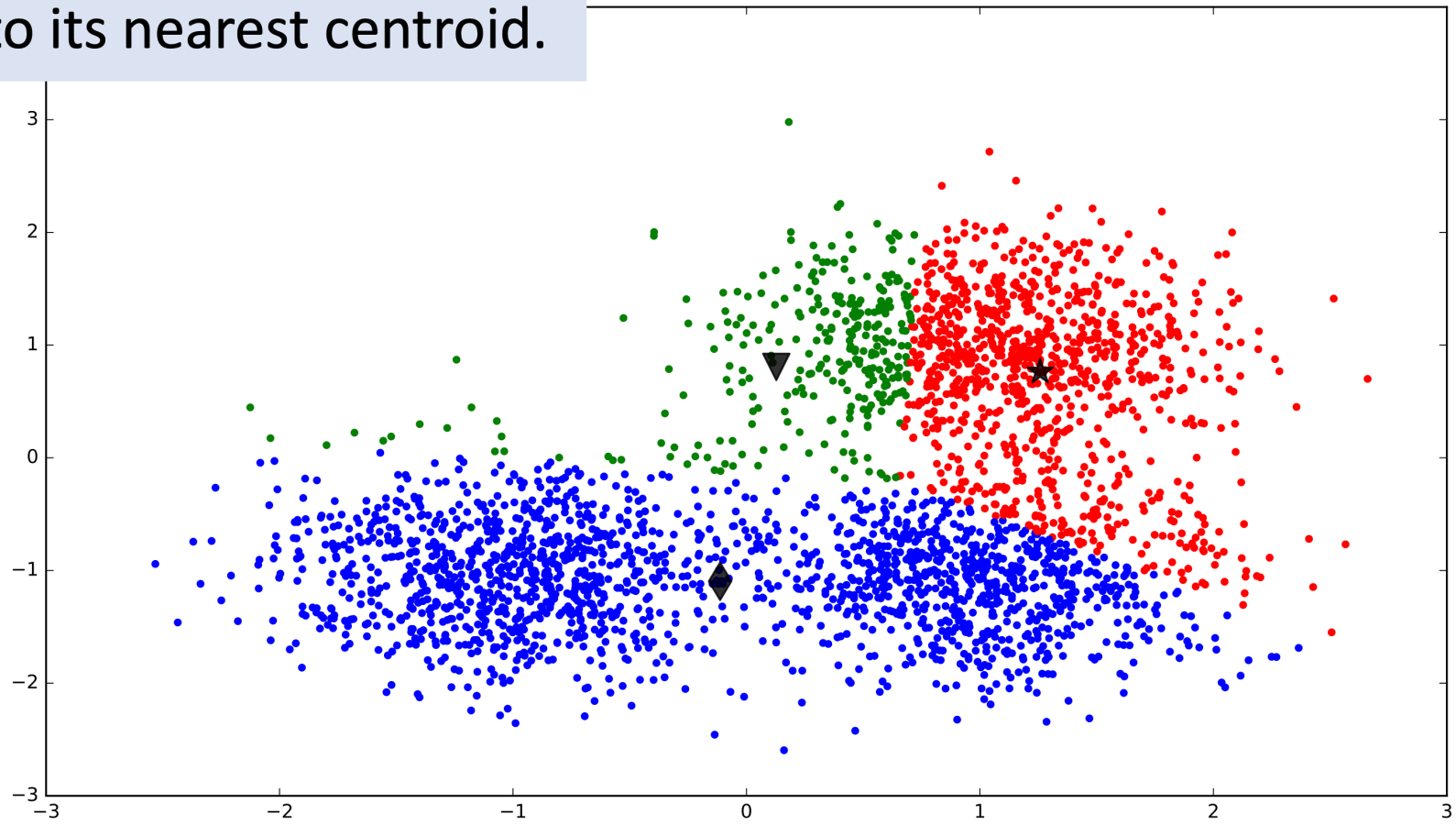



Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Lloyd's Algorithm

Assign $\mathbf{x}_j$ to its nearest centroid.

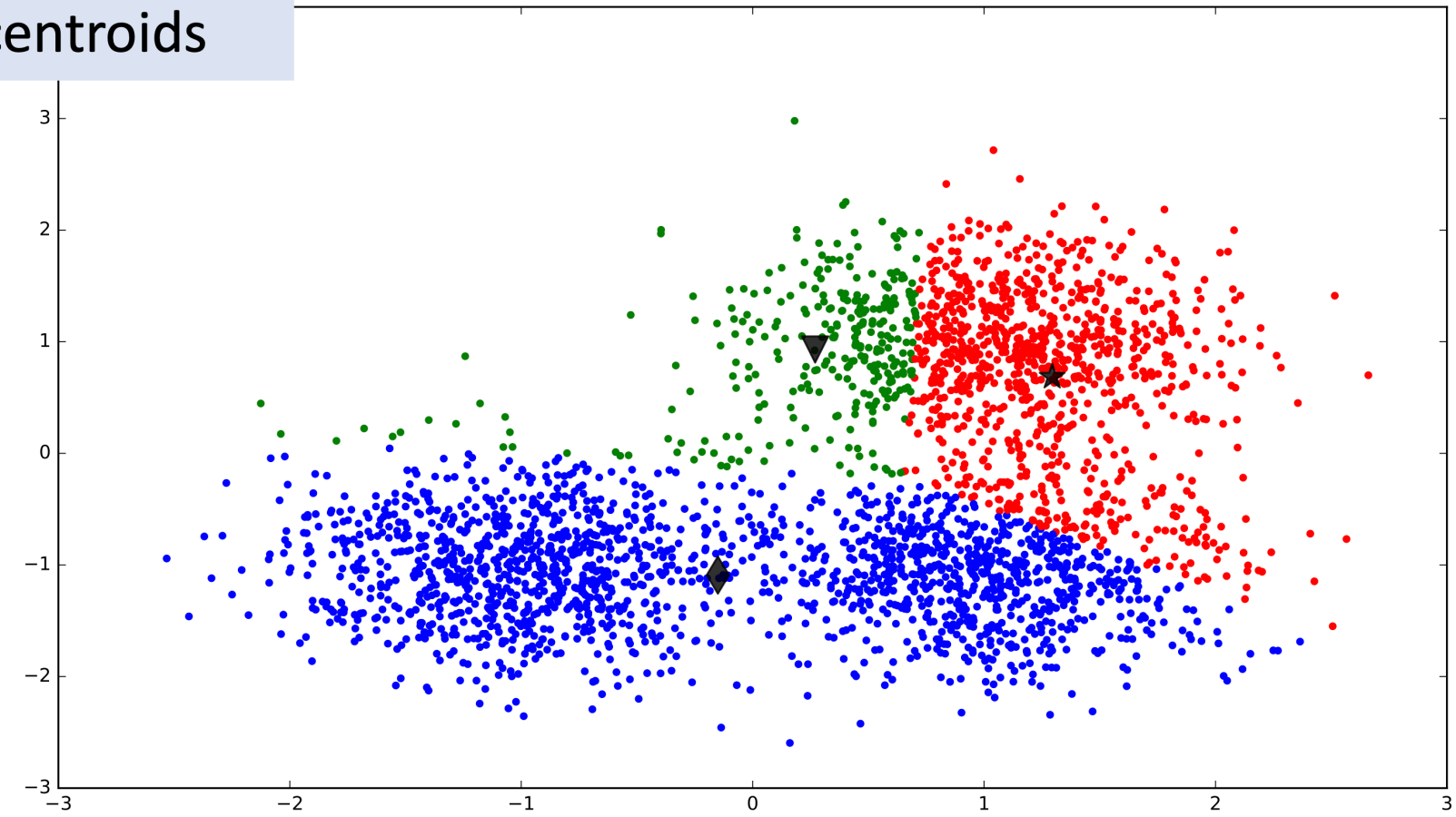



Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Lloyd's Algorithm

move centroids

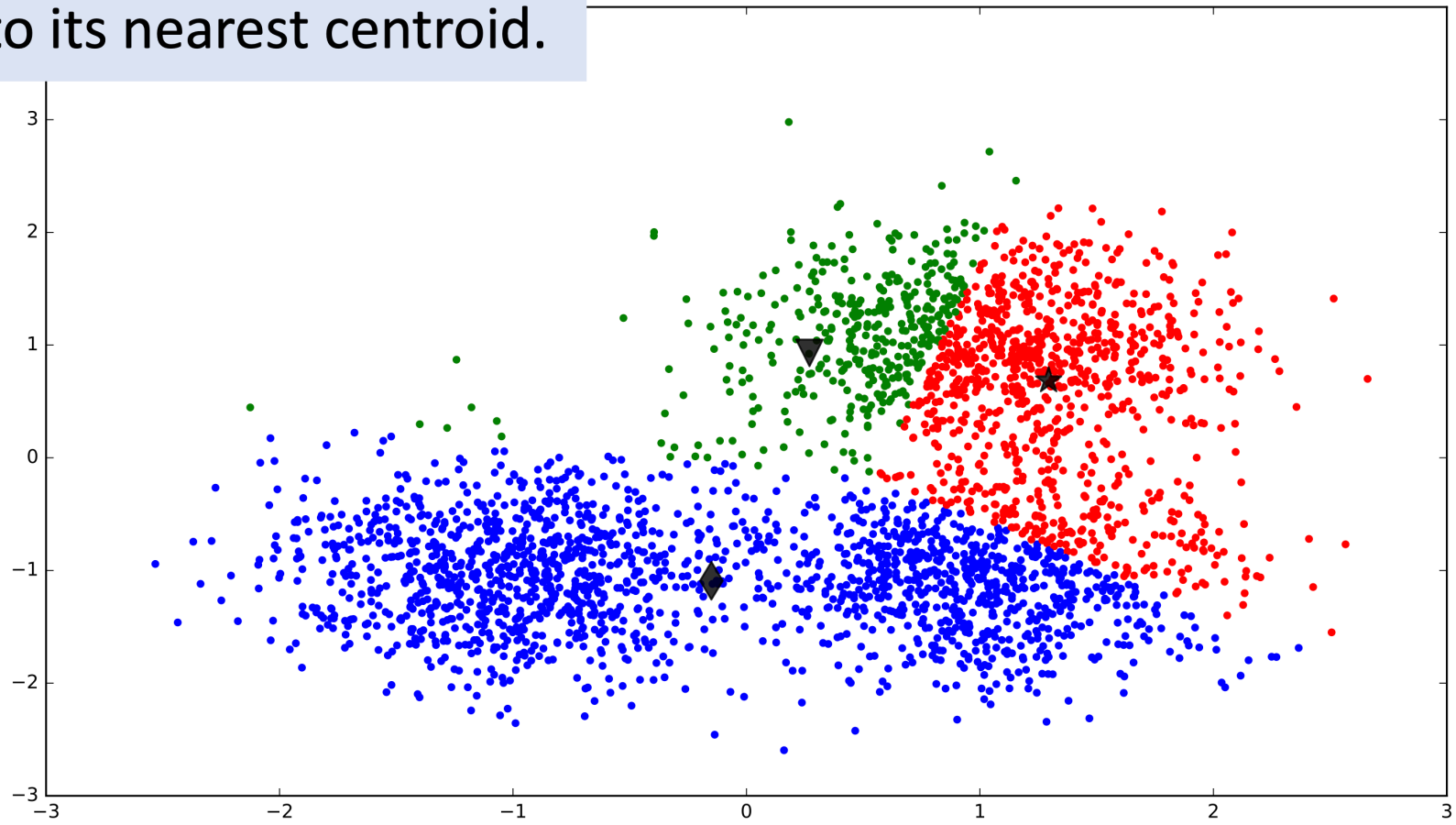



Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Lloyd's Algorithm

Assign $\mathbf{x}_j$ to its nearest centroid.

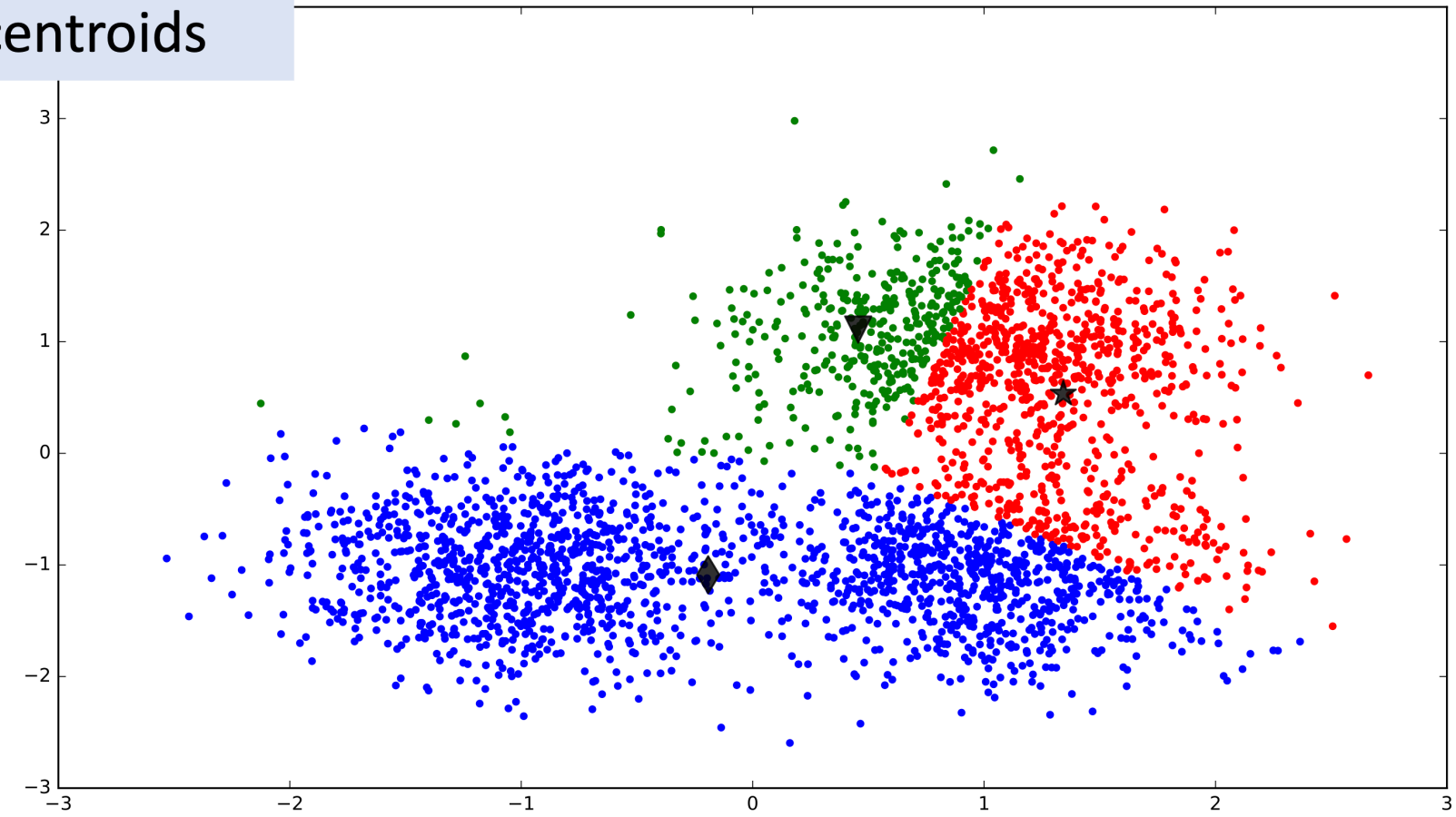



Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Lloyd's Algorithm

move centroids

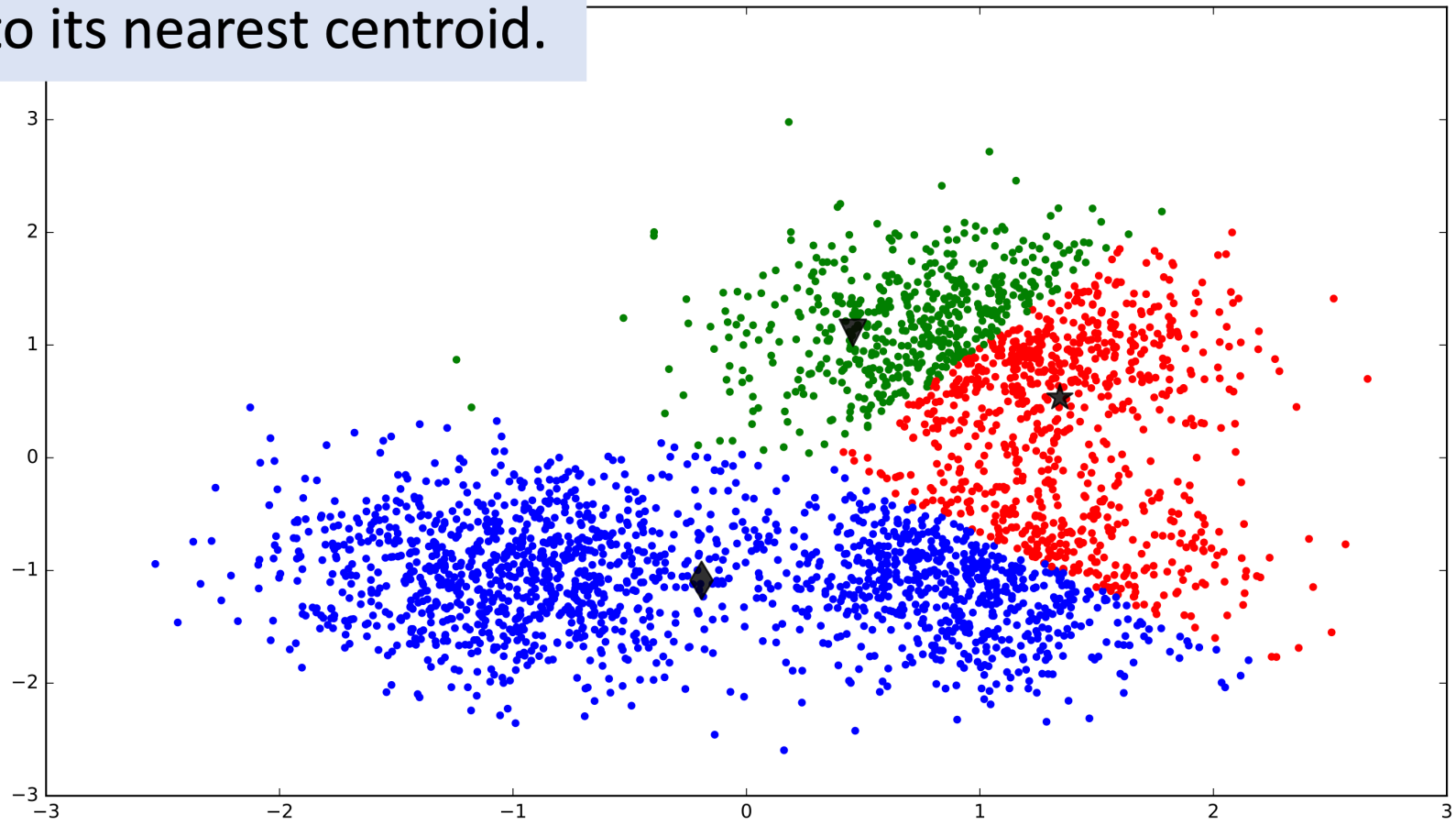



Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Lloyd's Algorithm

Assign $\mathbf{x}_j$ to its nearest centroid.

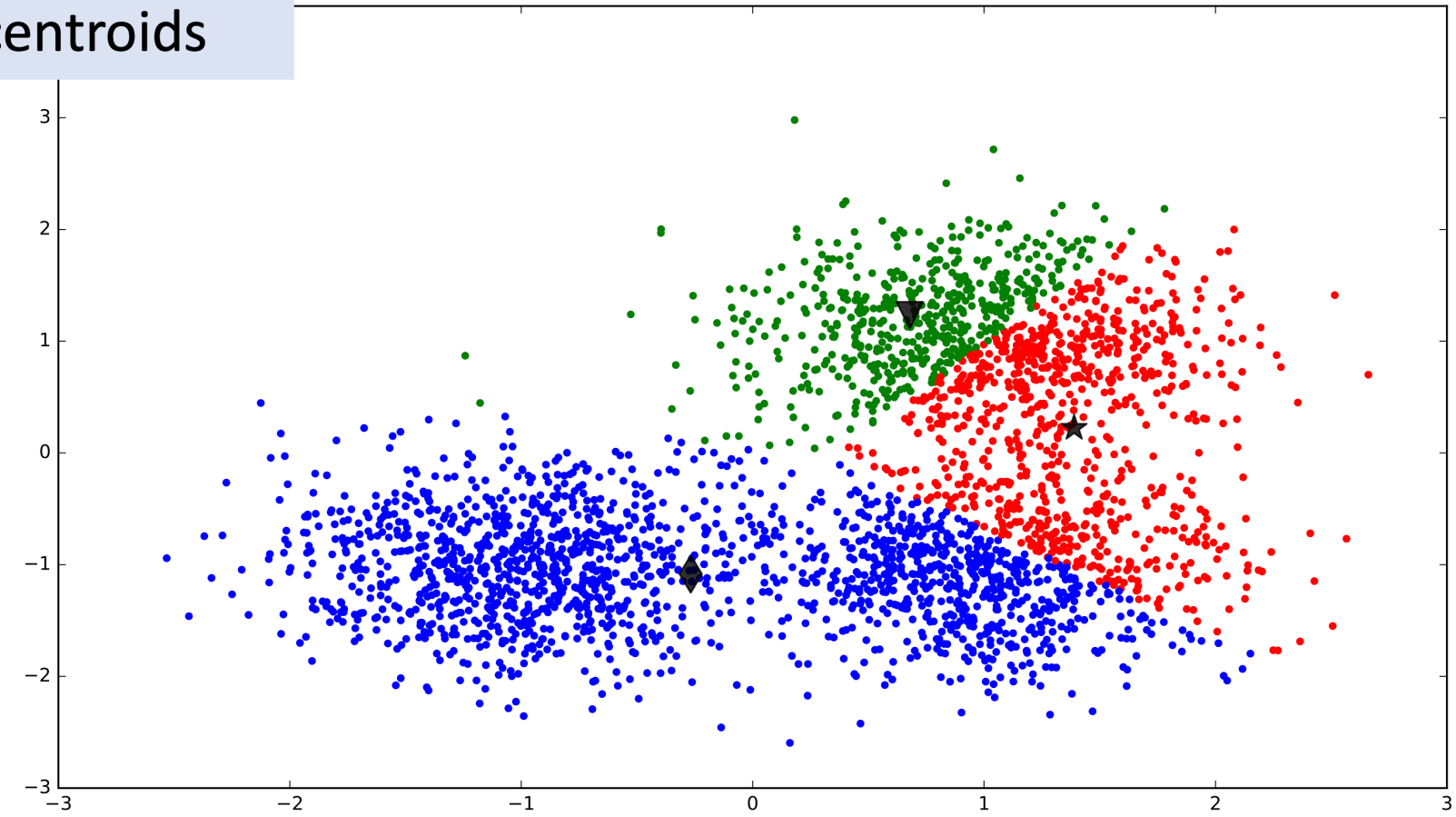



Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Lloyd's Algorithm

move centroids

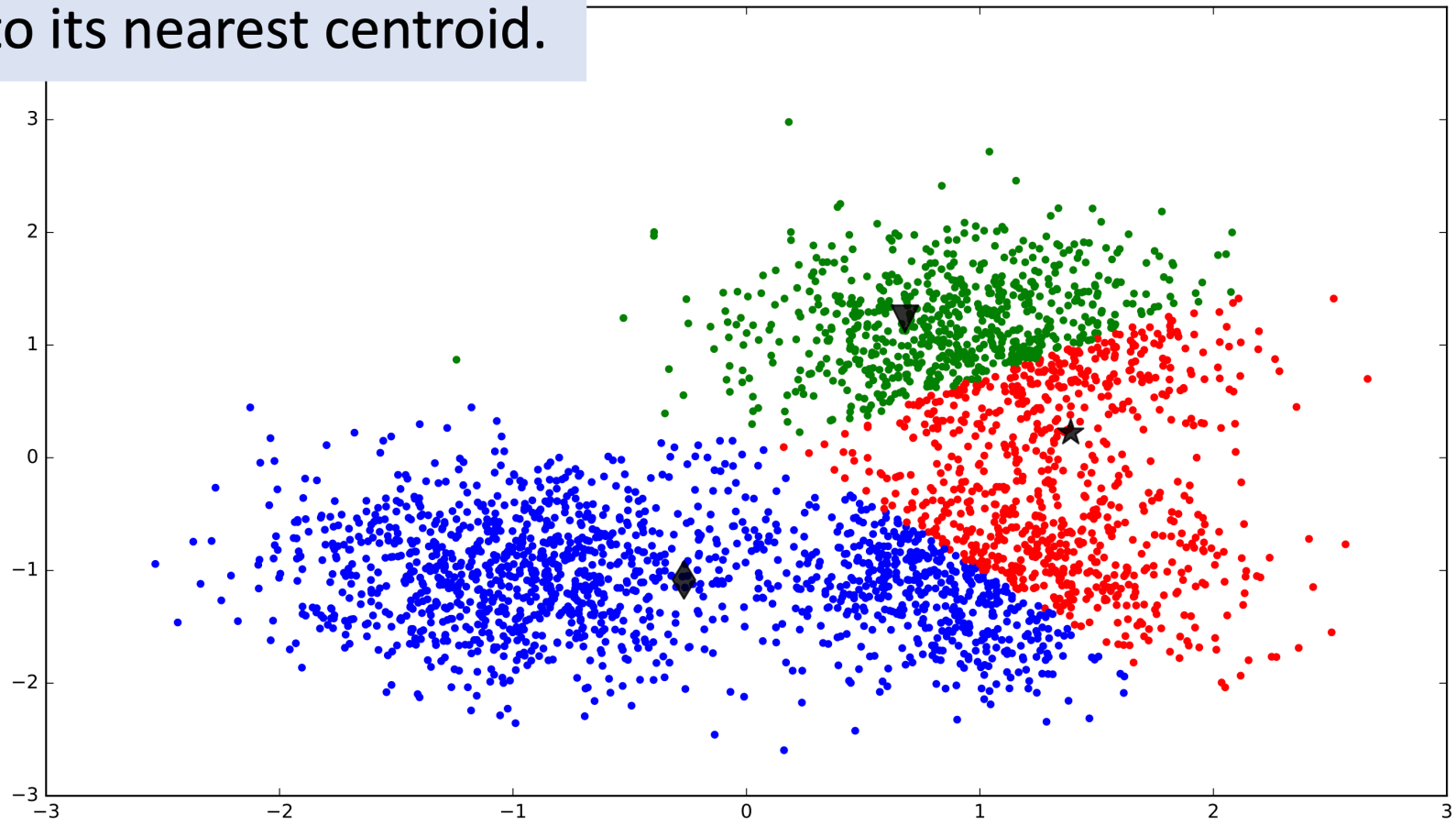



Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Lloyd's Algorithm

Assign $\mathbf{x}_j$ to its nearest centroid.

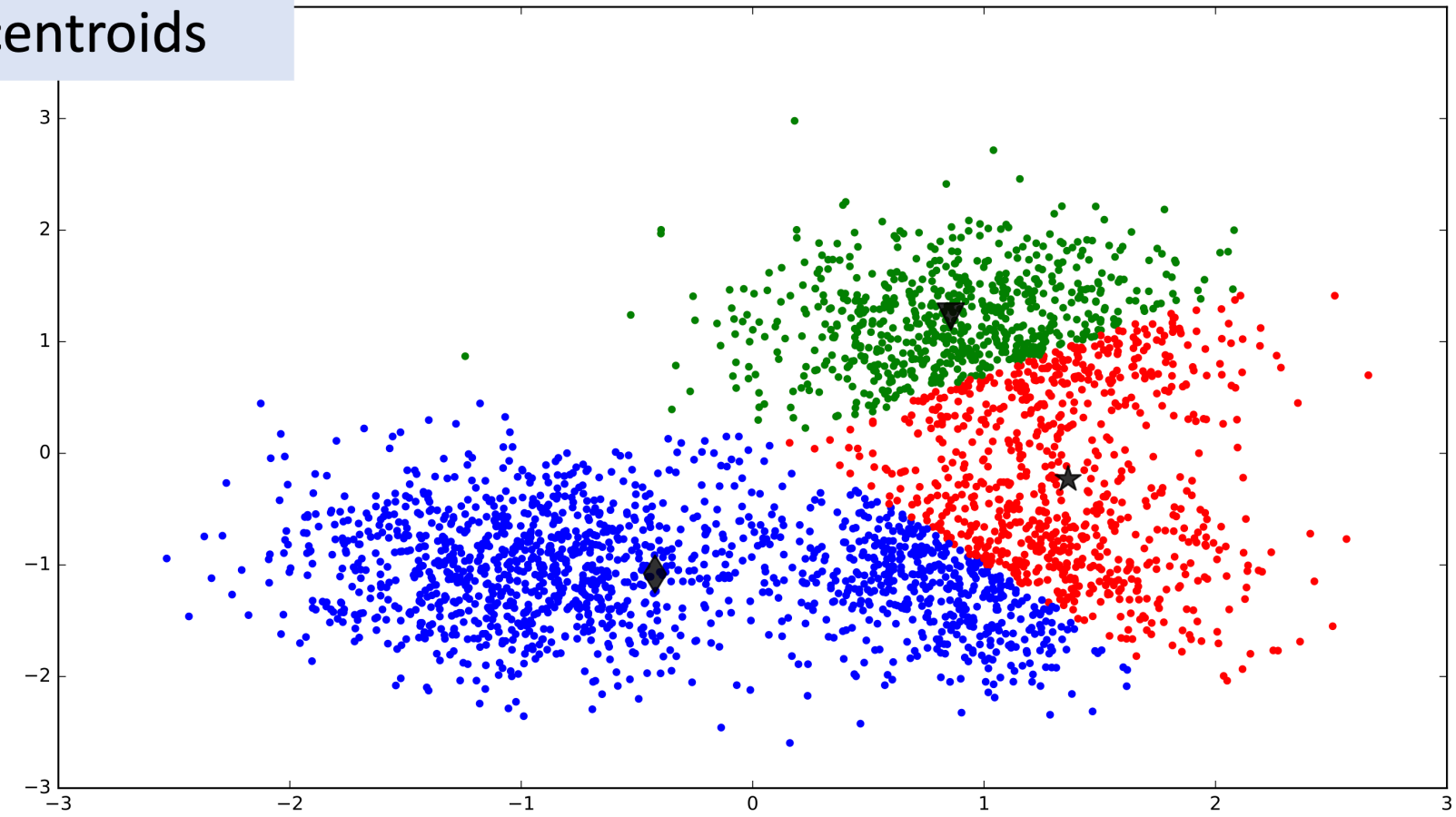



Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Lloyd's Algorithm

move centroids

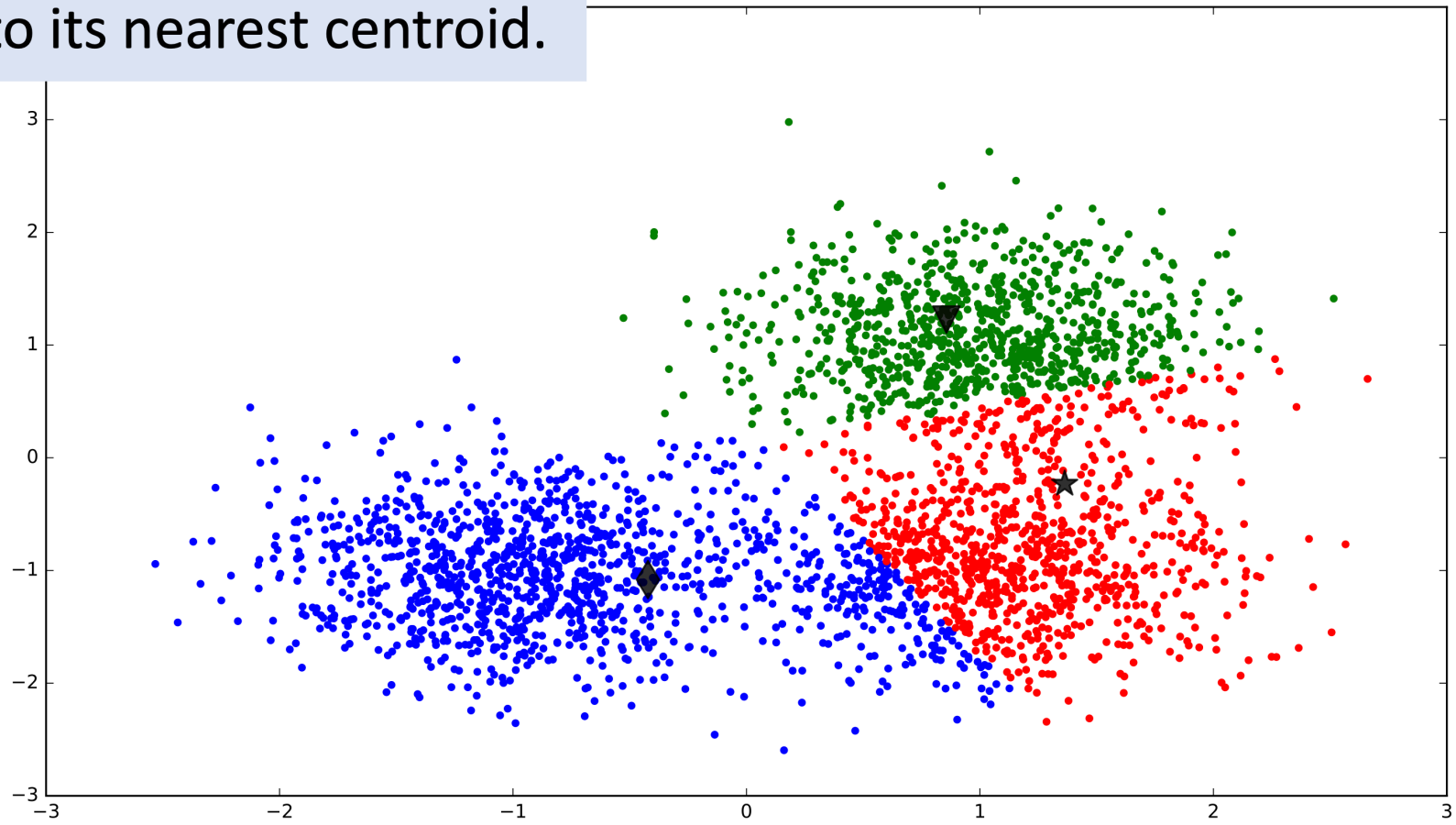



Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Lloyd's Algorithm

Assign $\mathbf{x}_j$ to its nearest centroid.

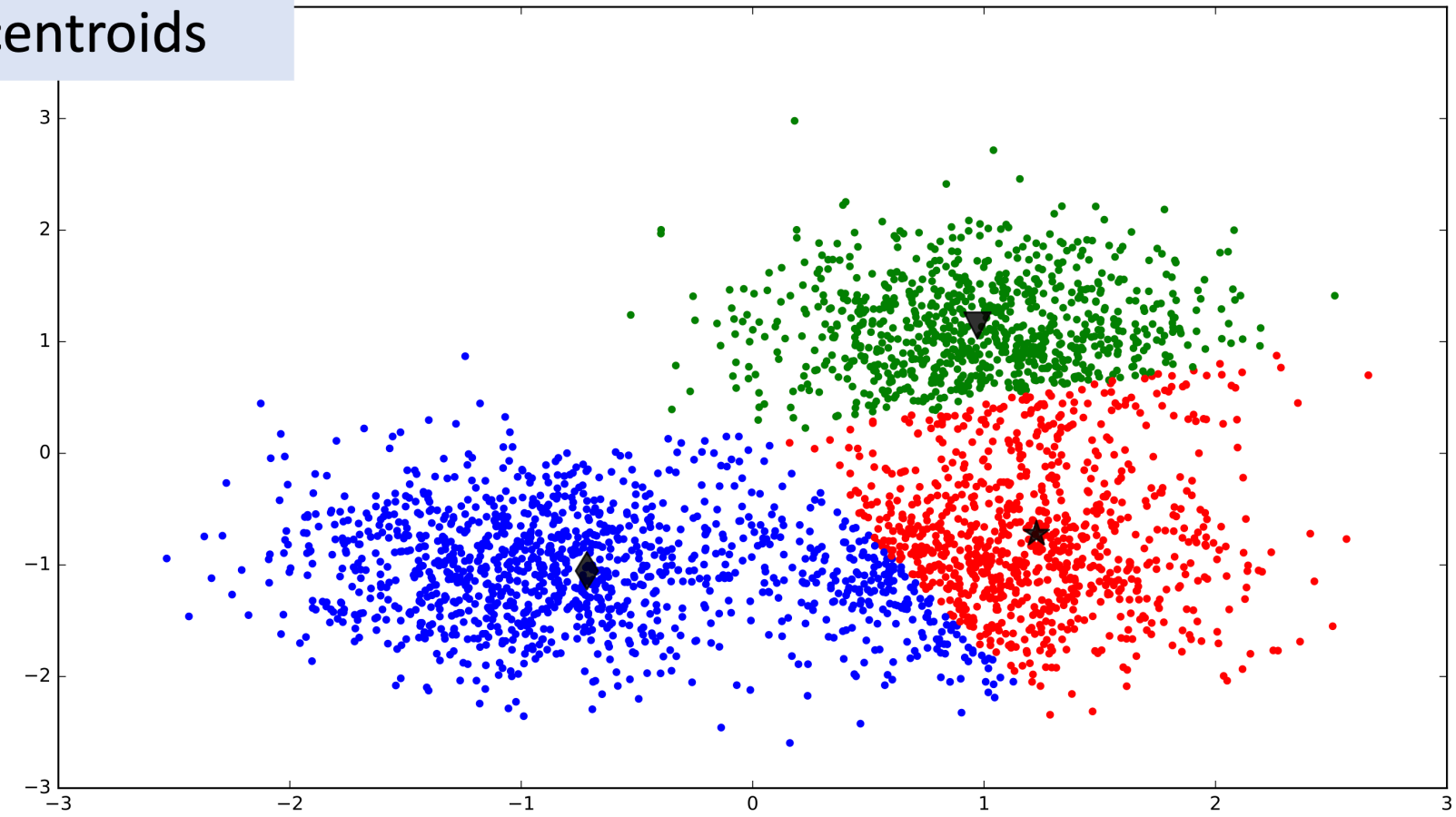



Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Lloyd's Algorithm

move centroids

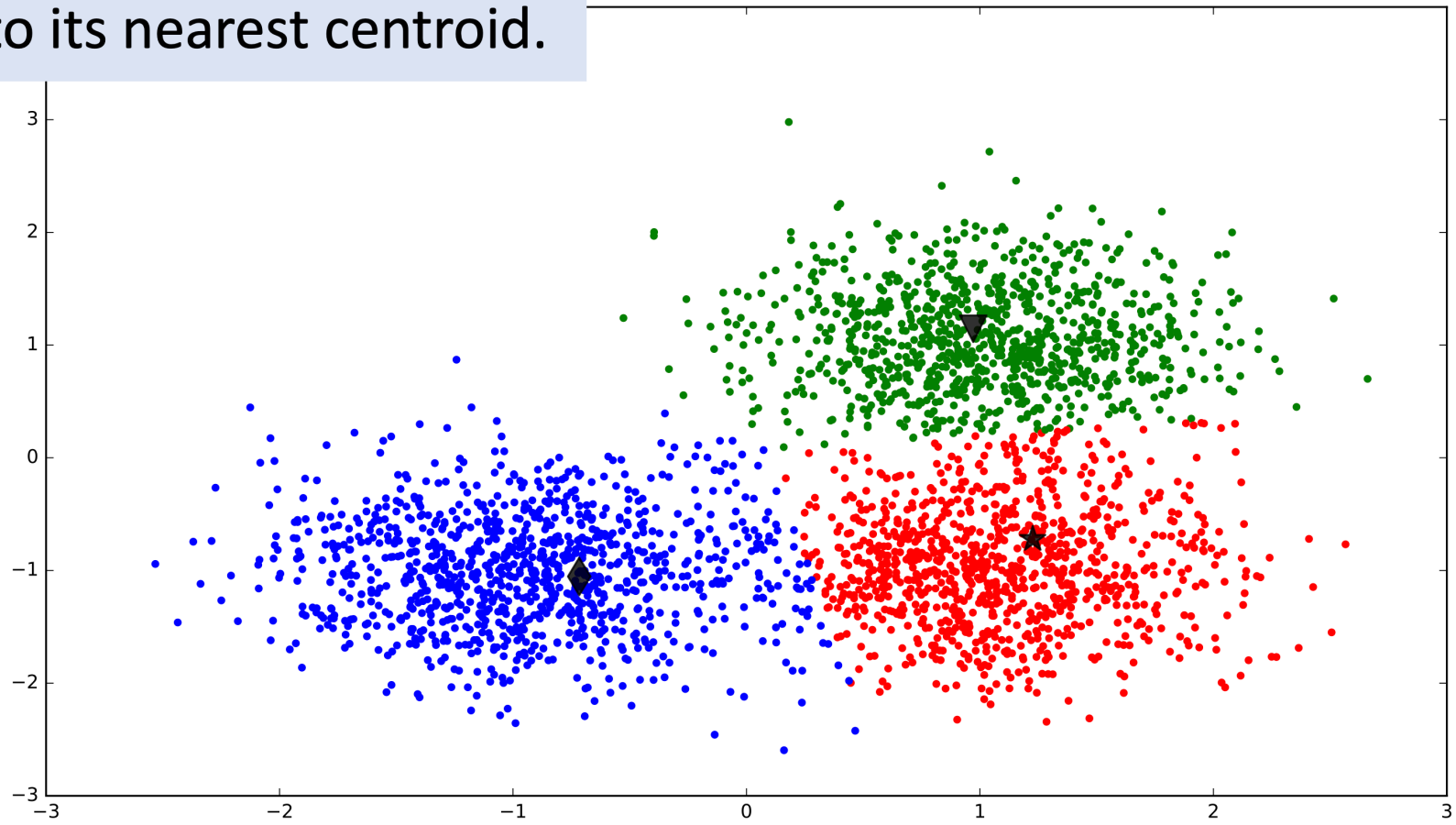



Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Lloyd's Algorithm

Assign $\mathbf{x}_j$ to its nearest centroid.

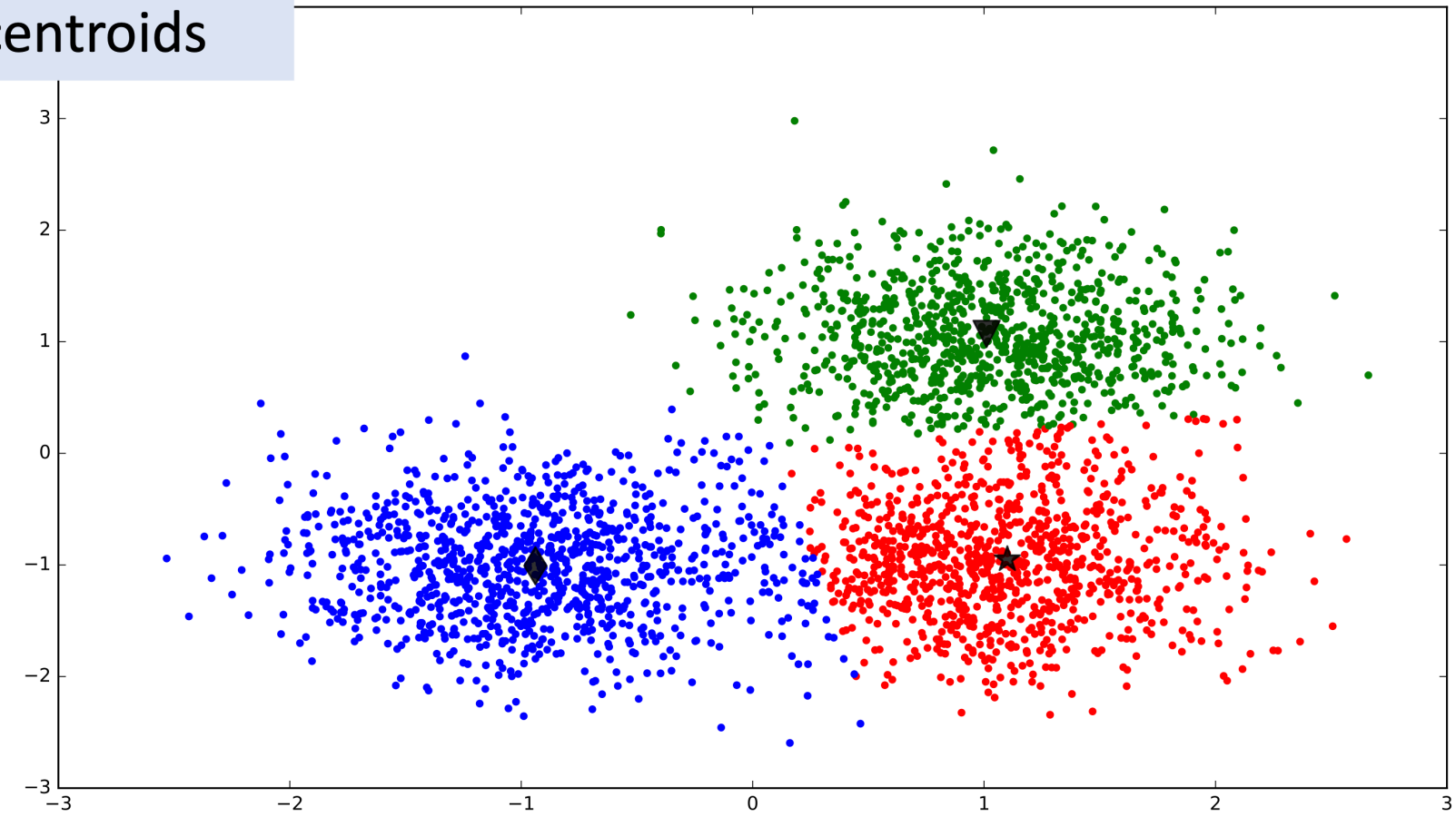



Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Lloyd's Algorithm

move centroids

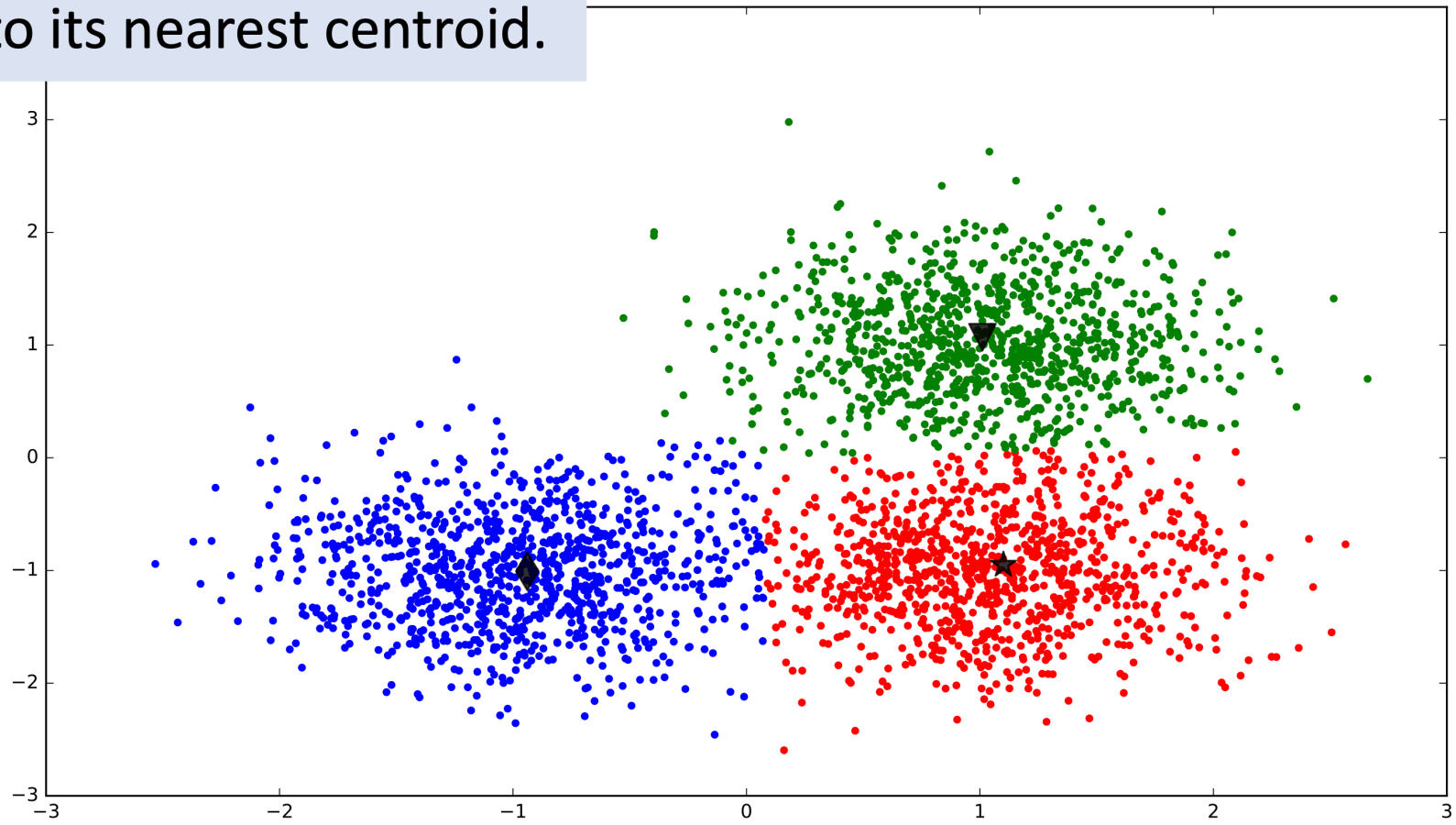



Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Lloyd's Algorithm

Assign $\mathbf{x}_j$ to its nearest centroid.



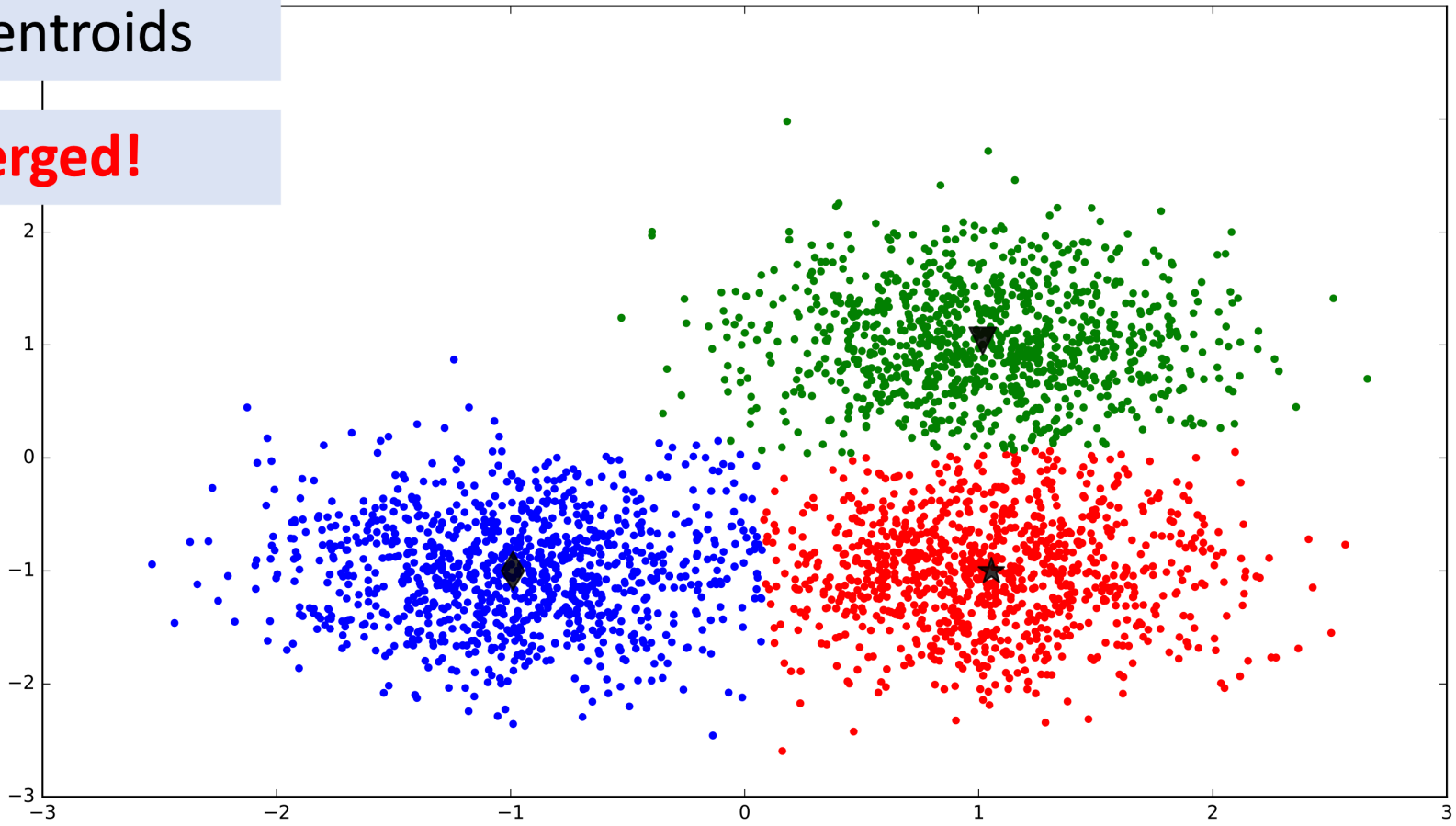

Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Lloyd's Algorithm

move centroids

converged!




Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Summary

- Clustering task: partition $[n]$ into $k$ subsets according to the feature vectors.
- K-means model: $\min_{S_1,\cdots,S_k} \sum_{i=1}^{k} \sum_{j \in S_i} \left\| \mathbf{x}_j - \frac{1}{|S_i|} \sum_{l \in S_i} \mathbf{x}_l \right\|_2^2$.
- Lloyd's algorithm for solving the model (approximately).
- Other algorithms: Forgy, MacQueen, and Hartigan.

| Tasks | Methods | Algorithms |
|---|---|---|
| clustering | K-means | Lloyd's algorithm |

Shusen Wang

Slides courtesy of Dr. Shusen Wang

# Summary

- Clustering task: partition $[n]$ into $k$ subsets according to the feature vectors.
- K-means model: $\min_{S_1,\cdots,S_k} \sum_{i=1}^{k} \sum_{j\in S_i} \left\| \mathbf{x}_j - \frac{1}{|S_i|} \sum_{l\in S_i} \mathbf{x}_l \right\|_2^2$.
- Lloyd's algorithm for solving the model (approximately).
- Other algorithms: Forgy, MacQueen, and Hartigan.

- There is not such a thing called "k-means algorithm"!

Slides courtesy of Dr. Shusen Wang

# MobileNet

# MobileNet

- Motivations
  - Small enough to fit in an edge computing device.
  - Less computation than the standard ConvNets.

- Key idea: Depthwise separable convolution.

- Paper: https://arxiv.org/pdf/1704.04861.pdf

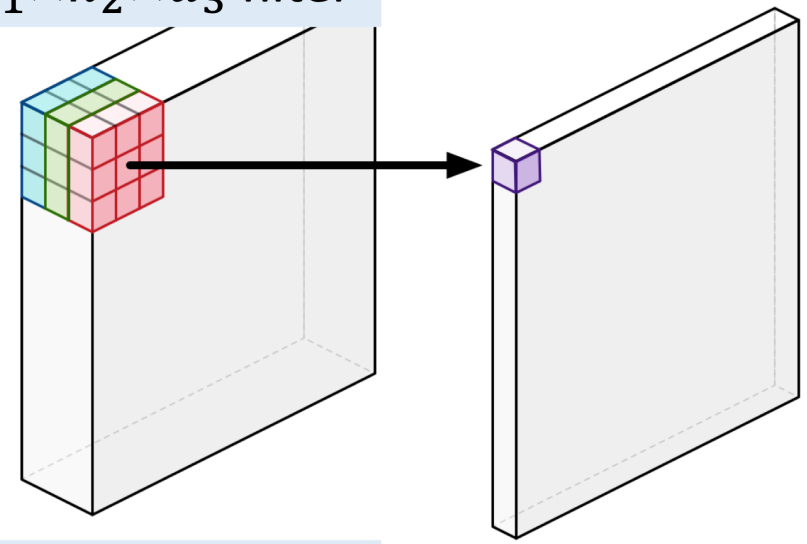
- Further reading:
  - https://machinethink.net/blog/googles-mobile-net-architecture-on-iphone/
  - https://medium.com/@yu4u/why-mobilenet-and-its-variants-e-g-shufflenet-are-fast-1c7048b9618d

# Convolution

**convolution**

`keras.layers.Conv2D`

One $k_1 \times k_2 \times d_3$ filter



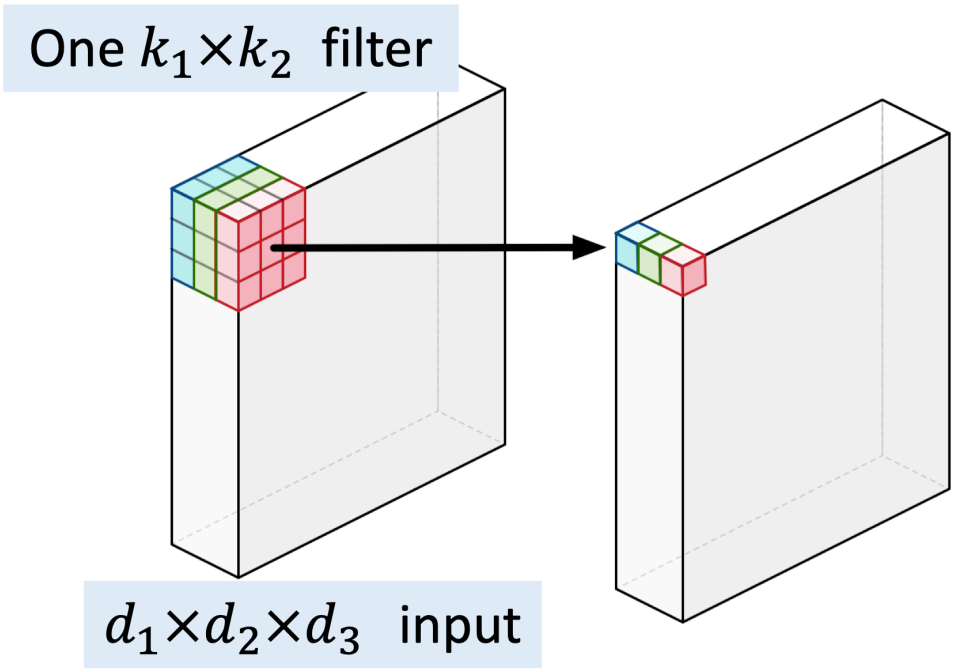
$d_1 \times d_2 \times d_3$ input

In this example, $d_3 = 3$.

Slides courtesy of Dr. Shusen Wang

# Convolution

**depthwise convolution**
`keras.layers.DepthwiseConv2D`



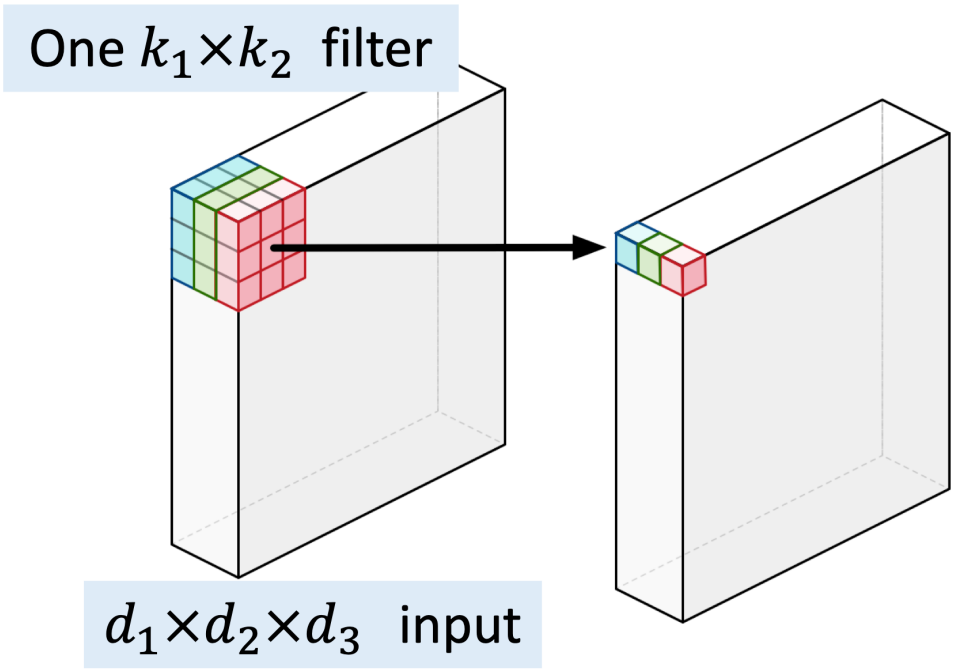
# Convolution

**depthwise convolution**

`keras.layers.DepthwiseConv2D`

For $i = 1$ to $d_3$ (each of the slices):

- Compute the convolution of the $i$-th slice (shape $d_1 \times d_2$) and the filter (shape $k_1 \times k_2$).
- Output: a matrix (shape $d_1 \times d_2$).



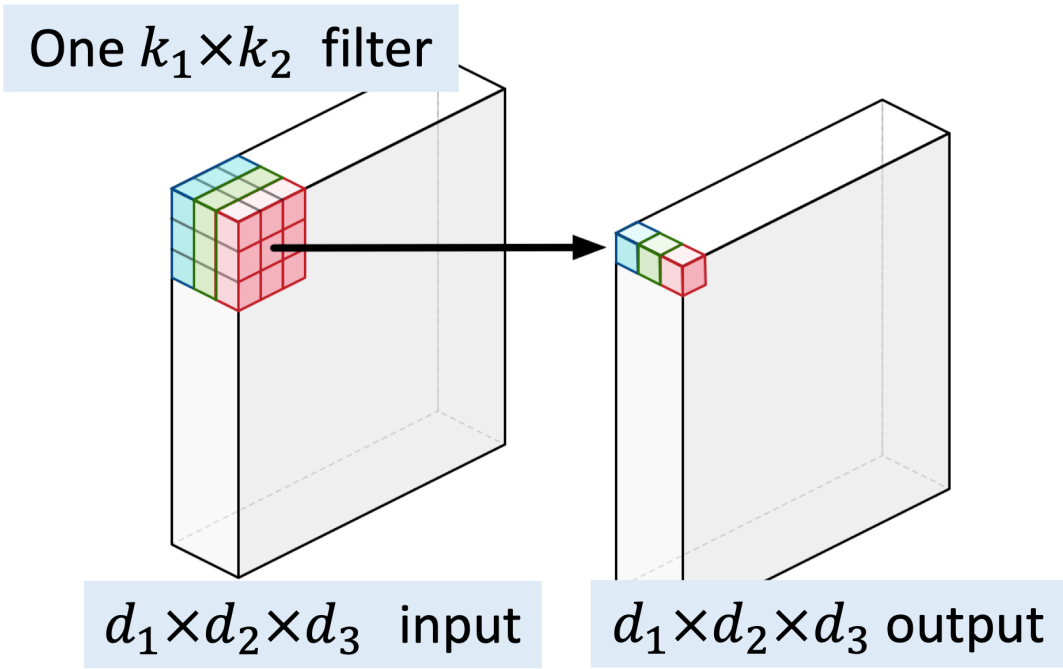
# Convolution

**depthwise convolution**
`keras.layers.DepthwiseConv2D`

For $i = 1$ to $d_3$ (each of the slices):

- Compute the convolution of the $i$-th slice (shape $d_1 \times d_2$) and the filter (shape $k_1 \times k_2$).
- Output: a matrix (shape $d_1 \times d_2$).

Thus the final output is $d_1 \times d_2 \times d_3$.

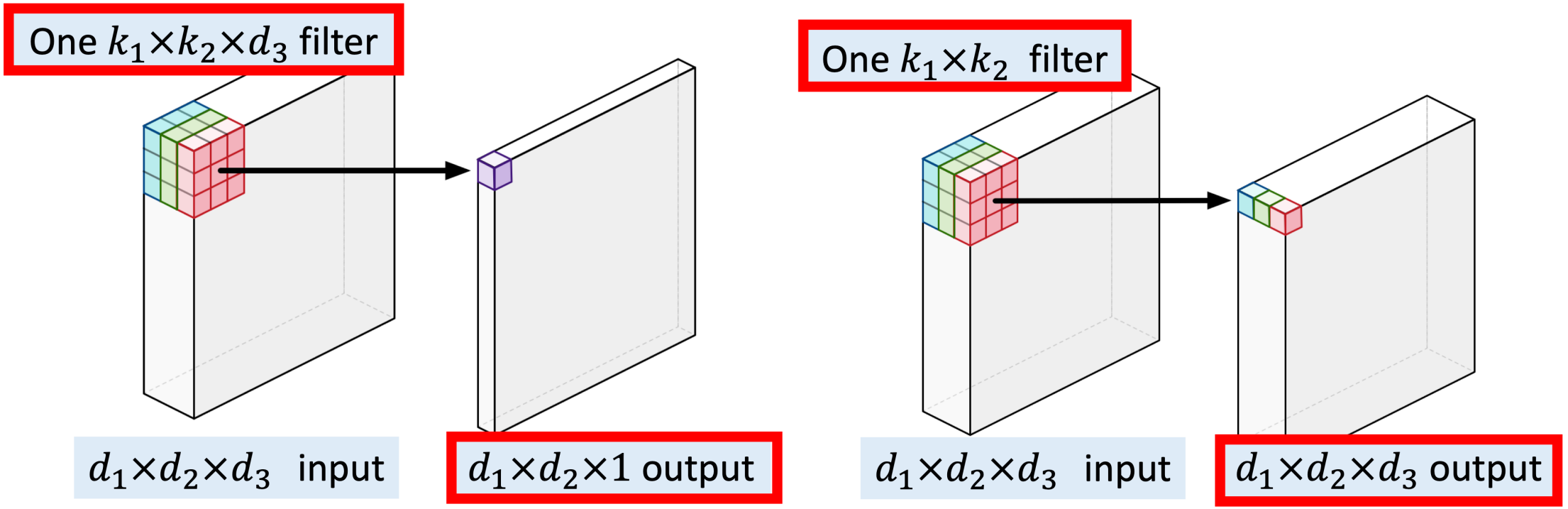



Slides courtesy of Dr. Shusen Wang

# Convolution

**convolution**
`keras.layers.Conv2D`

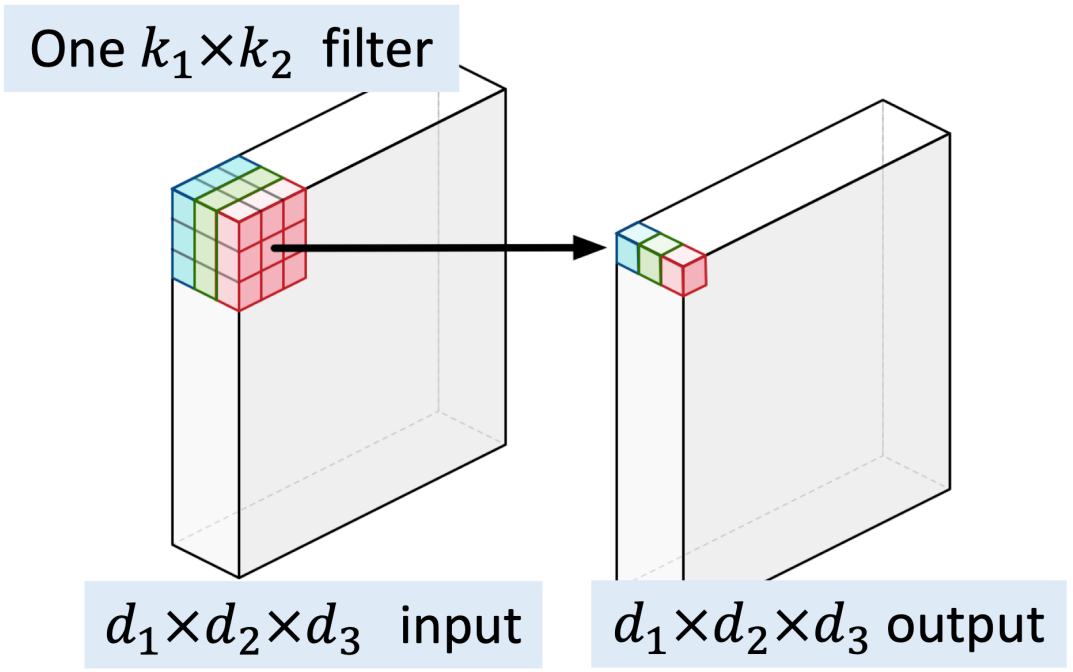
**depthwise convolution**
`keras.layers.DepthwiseConv2D`



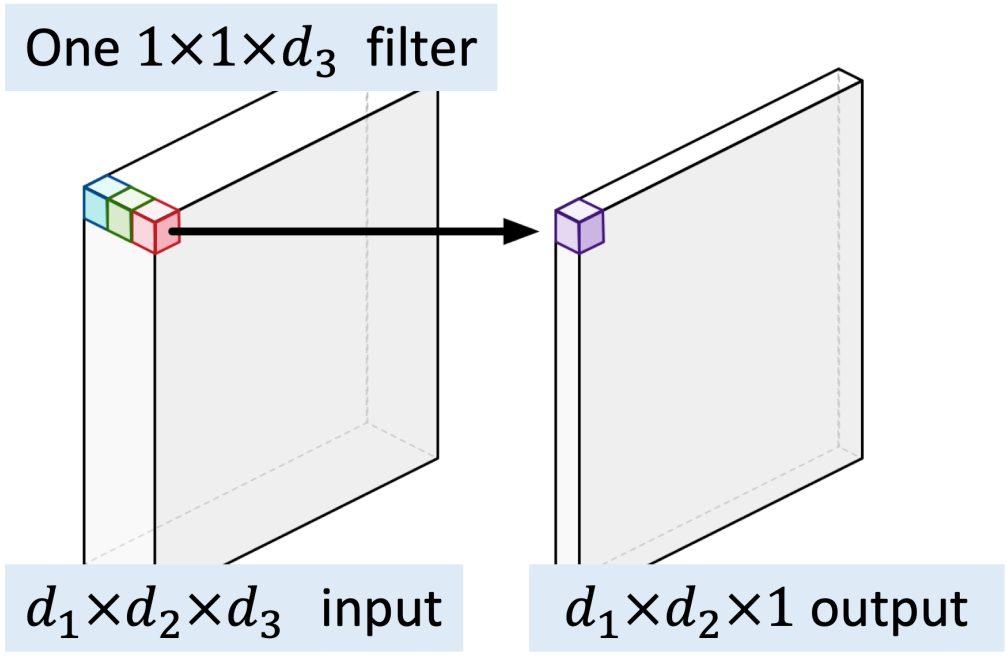

Slides courtesy of Dr. Shusen Wang

# MobileNet: Depthwise Conv + 1×1 Conv

**depthwise convolution**
`keras.layers.DepthwiseConv2D`

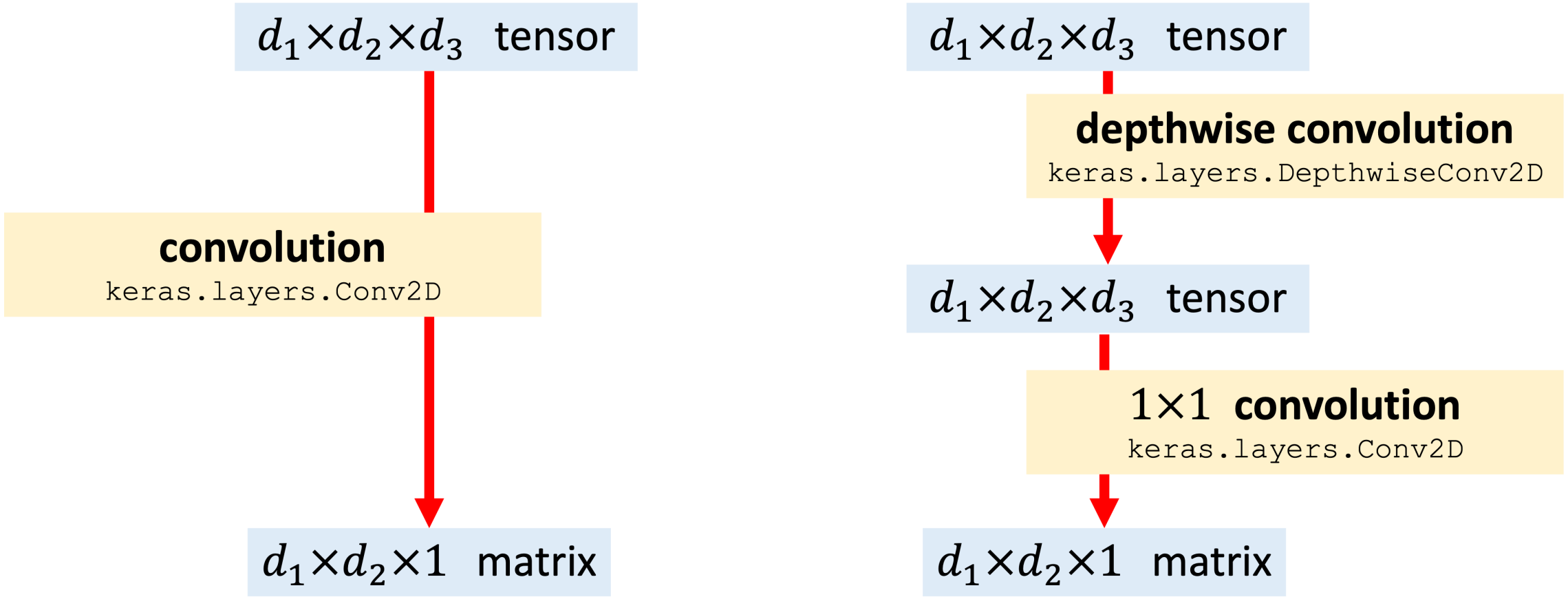



**1×1 convolution**
`keras.layers.Conv2D`



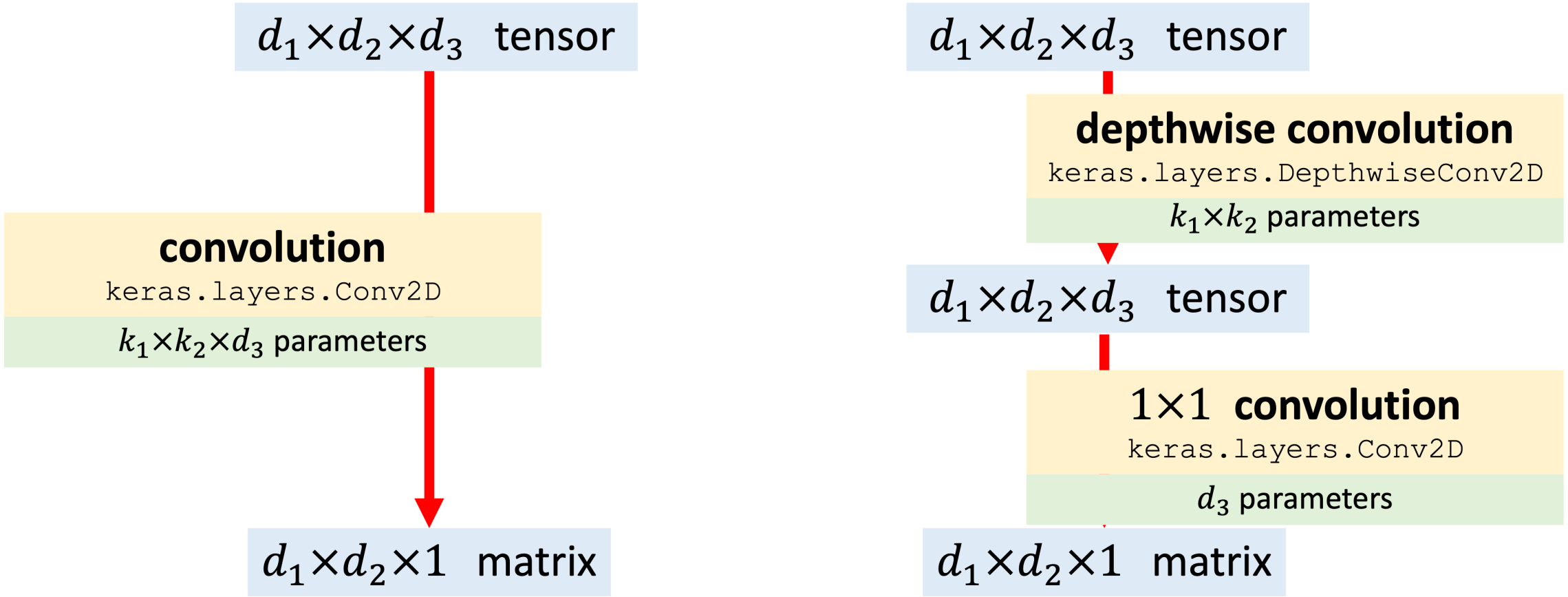

Slides courtesy of Dr. Shusen Wang

# MobileNet: Depthwise Separable Conv + 1×1 Conv



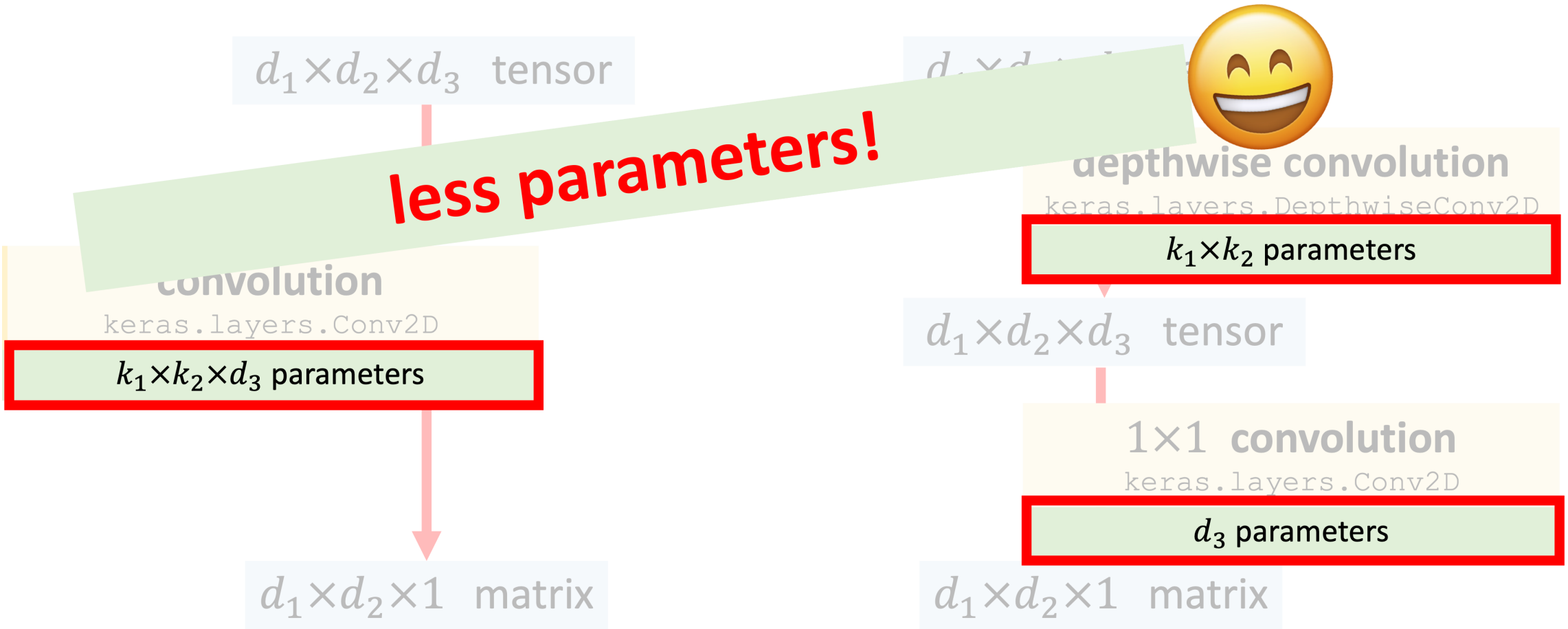

Slides courtesy of Dr. Shusen Wang

# MobileNet: Depthwise Separable Conv + 1×1 Conv



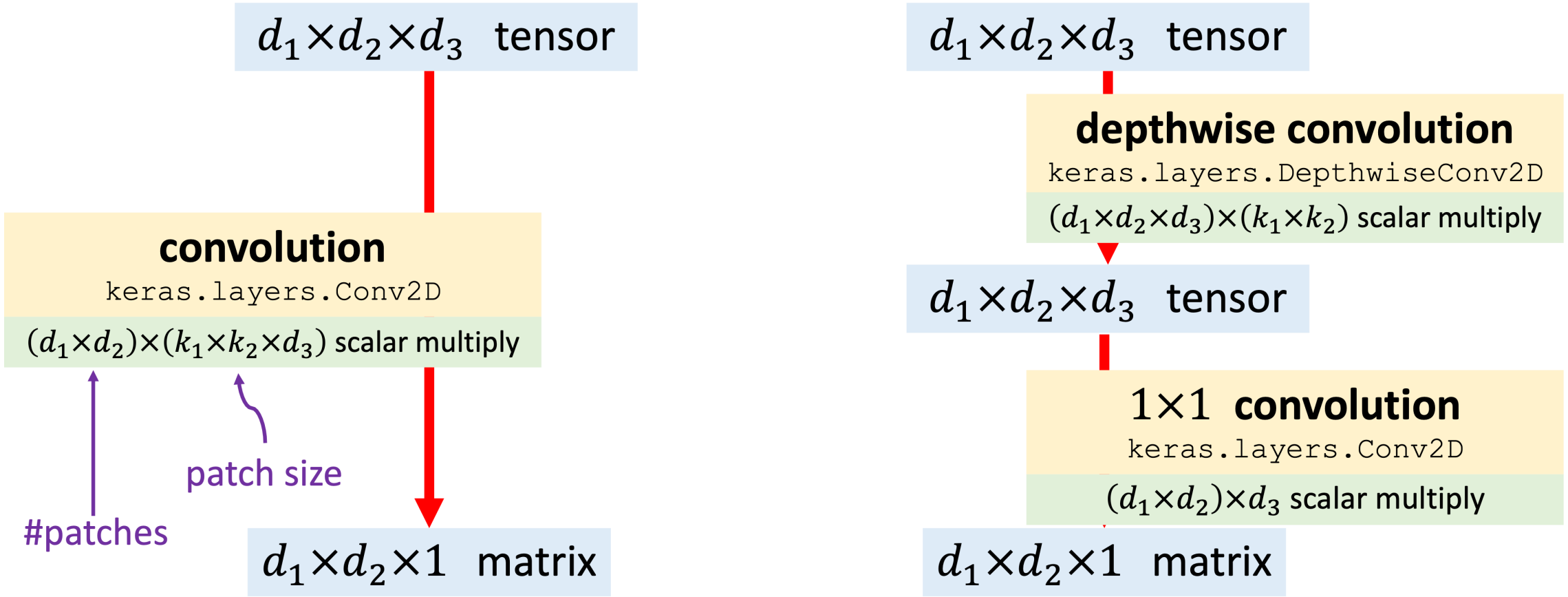

Slides courtesy of Dr. Shusen Wang

# MobileNet: Depthwise Separable Conv + 1×1 Conv



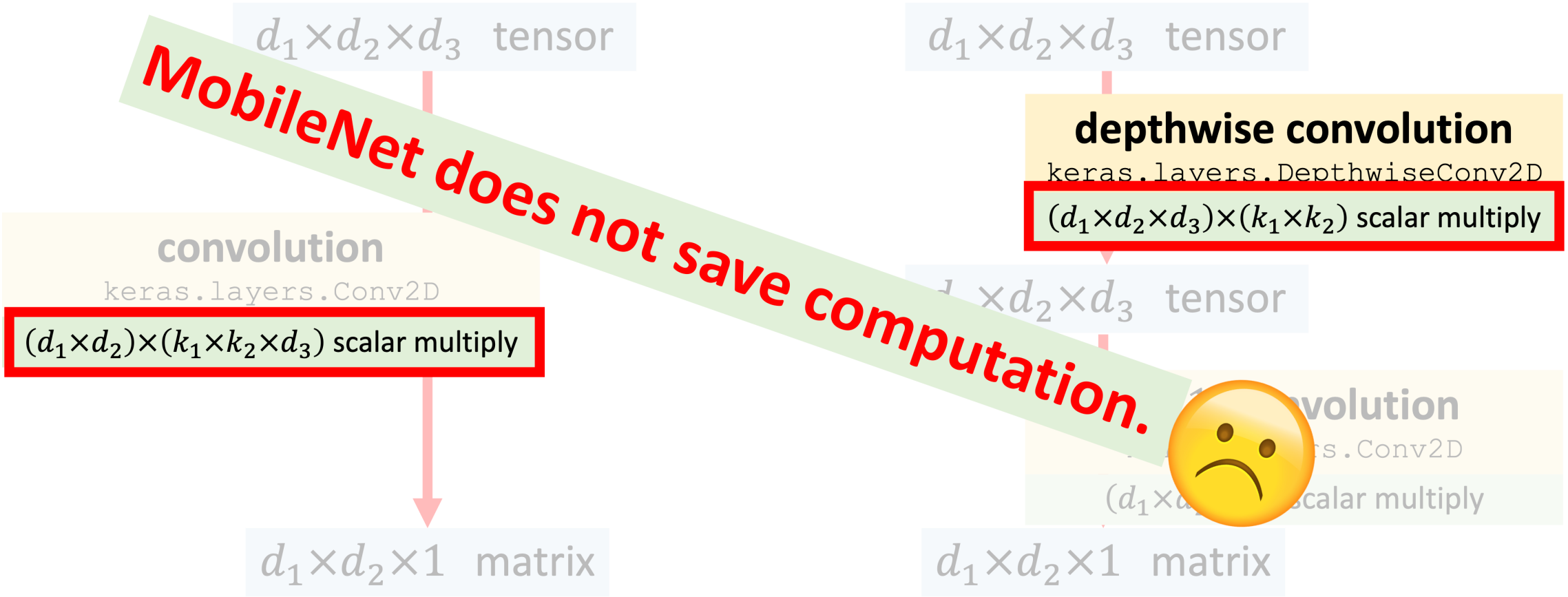

Slides courtesy of Dr. Shusen Wang

# MobileNet: Depthwise Separable Conv + 1×1 Conv

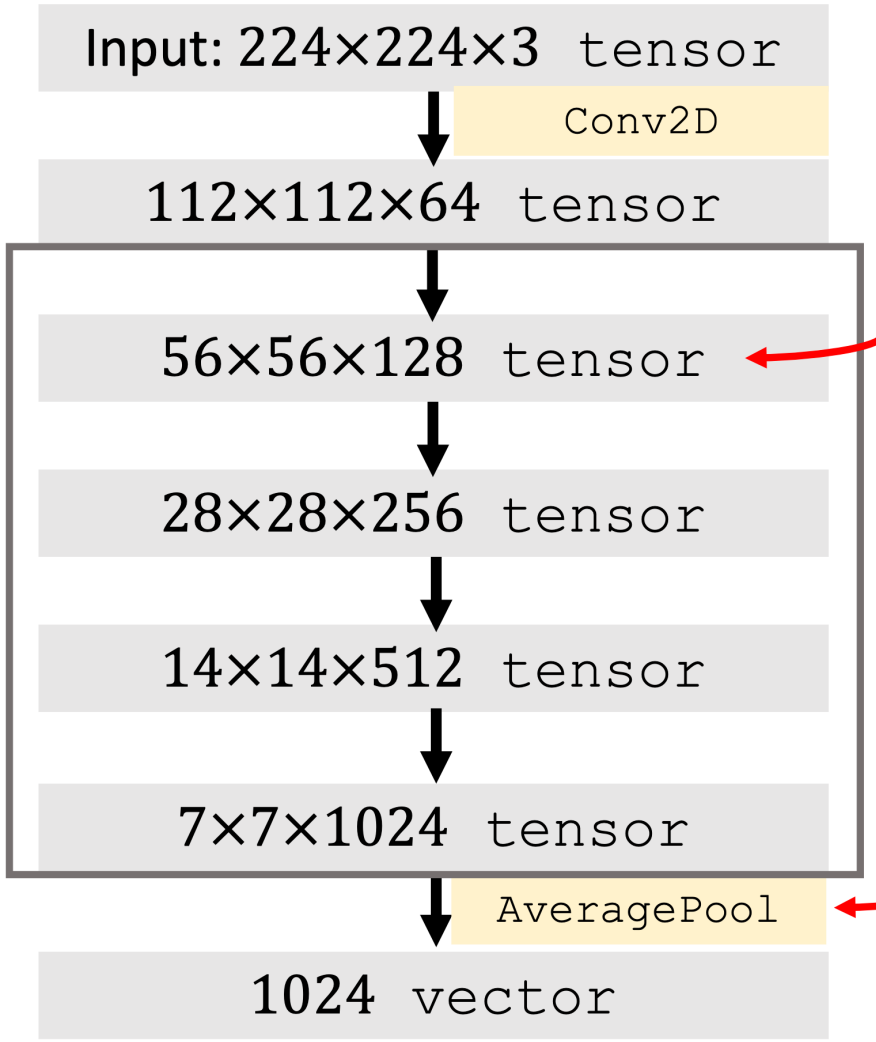



Slides courtesy of Dr. Shusen Wang

# MobileNet: Depthwise Separable Conv + 1×1 Conv



# MobileNet




Half the width and height, double the depth.

Why not using “Flatten”?

Slides courtesy of Dr. Shusen Wang

# Implementation in Keras

**separable convolution**
`keras.layers.SeparableConv2D`

- In Keras, you can directly use `SeparableConv2D` to build MobileNet.
- `SeparableConv2D = DepthwiseConv2D + Conv2D(1×1)`

**depthwise convolution**
`keras.layers.DepthwiseConv2D`

+

**1×1 convolution**
`keras.layers.Conv2D`

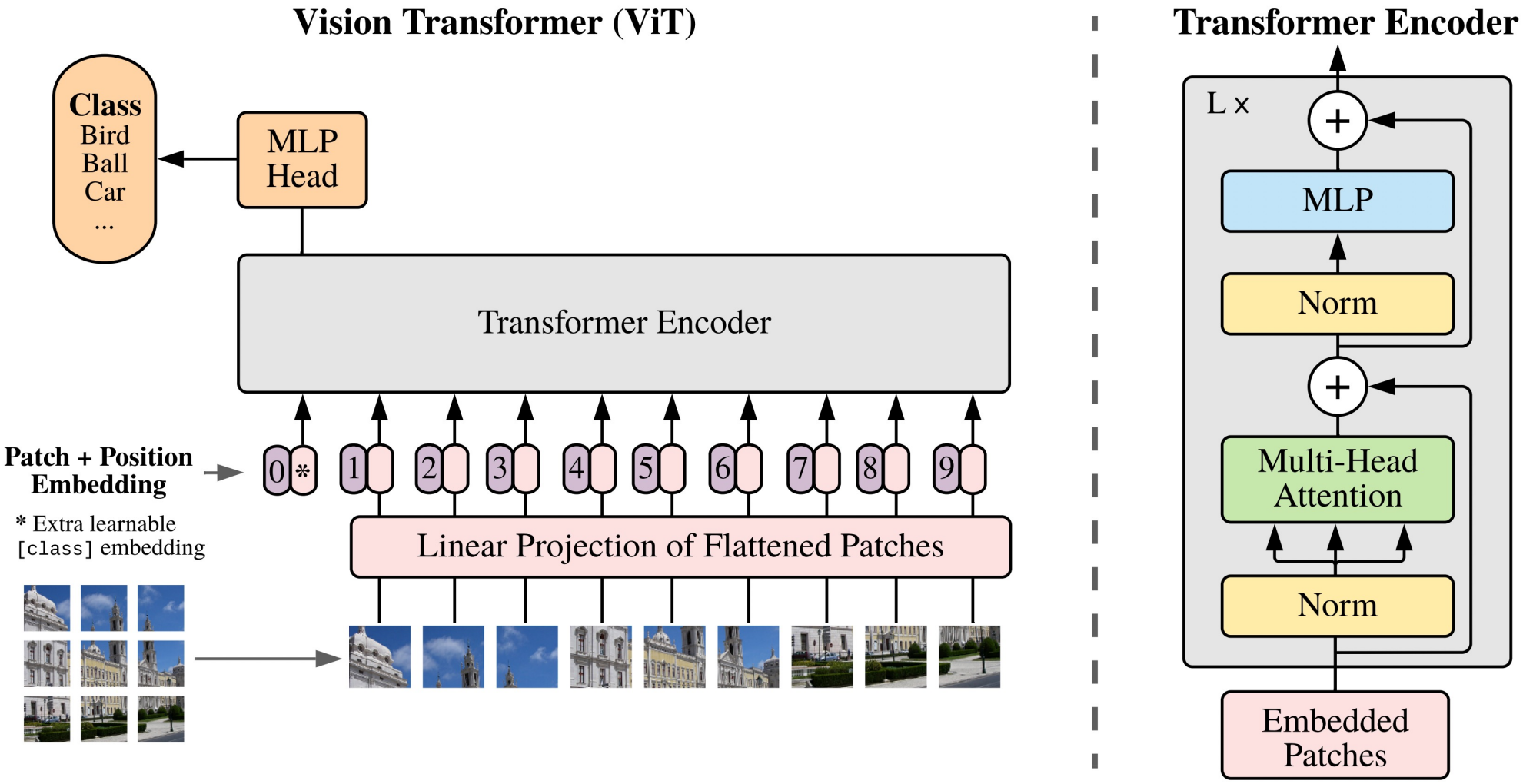
Slides courtesy of Dr. Shusen Wang

# Vision Transformer (ViT)

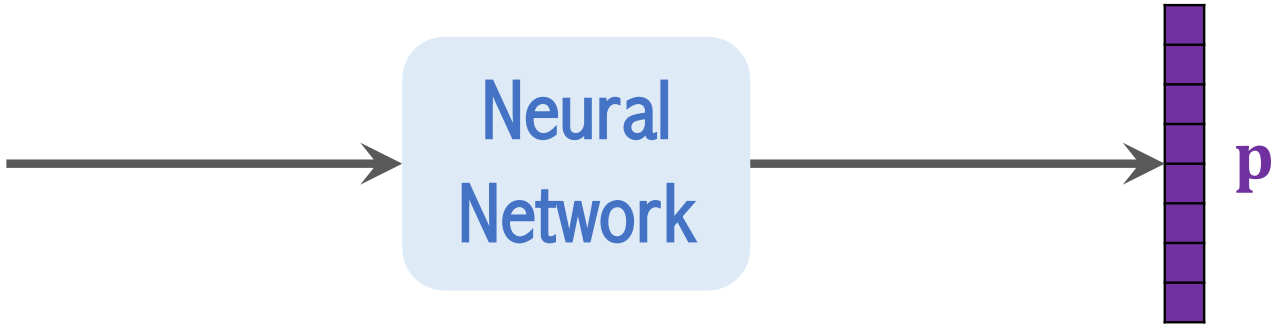
# Genealogy of Vision Transformer (ViT)

- RNN (before 1997)
- LSTM (before 2015)
  - Sepp Hochreiter; Jürgen Schmidhuber (1997). "Long short-term memory". Neural Computation. 9 (8): 1735–1780.
- Seq2Seq: LSTM model with attention (ICLR 2015)
  - Bahdanau, Cho, & Bengio. Neural machine translation by jointly learning to align and translate. In ICLR, 2015.
- Self-attention (EMNLP 2016)
  - Cheng, Dong, & Lapata. Long Short-Term Memory-Networks for Machine Reading. In EMNLP, 2016.
- Transformer (NeurIPS 2017)
  - Vaswani et al. Attention Is All You Need. In NIPS, 2017.
- Vision Transformer (ICLR 2021)
  - Dosovitskiy et al. An image is worth 16×16 words: transformers for image recognition at scale. In ICLR, 2021.

# Architecture of ViT

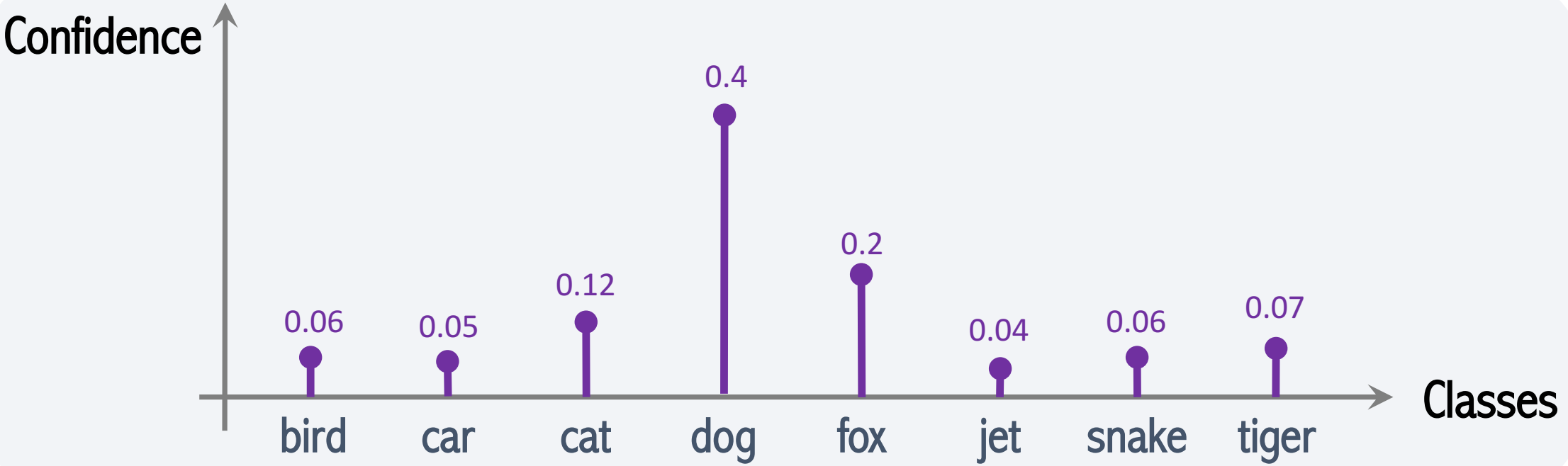


What is in the image?


Neural
Network
p

Neural Network

p




Slides courtesy of Dr. Shusen Wang

# Image Classification

- CNNs, e.g., ResNet, were the best solutions to image classification.
- Vision Transformer (ViT) [1] beats CNNs (by a small margin), if the dataset for pretraining is sufficiently large (at least 100 million images).
- ViT is based on Transformer (for NLP) [2].

**Reference**

1. Dosovitskiy et al. An image is worth 16×16 words: transformers for image recognition at scale. In *ICLR*, 2021.
2. Vaswani et al. Attention Is All You Need. In *NIPS*, 2017.

# Split Image into Patches

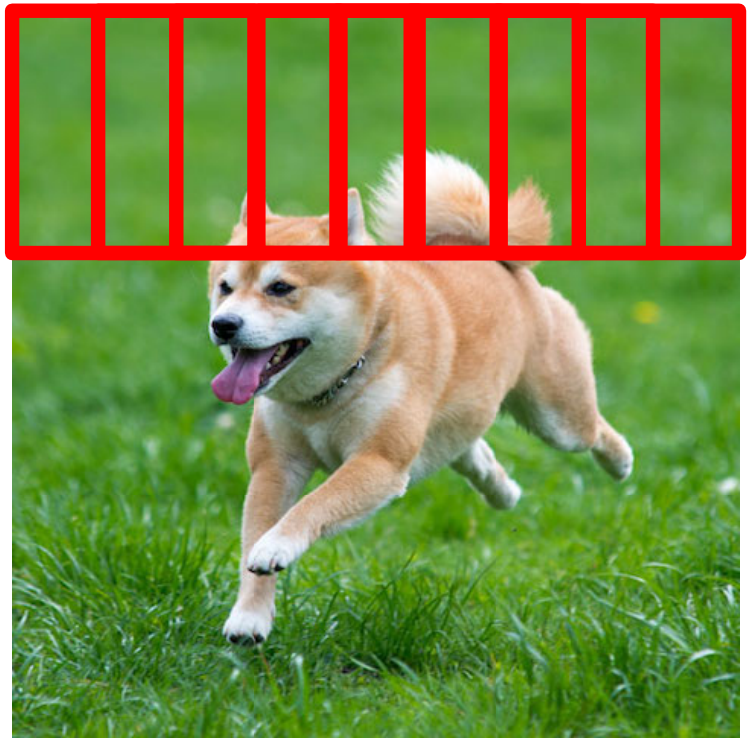
Slides courtesy of Dr. Shusen Wang

# Split Image into Patches

- Here, the patches do not overlap.

Slides courtesy of Dr. Shusen Wang

# Split Image into Patches

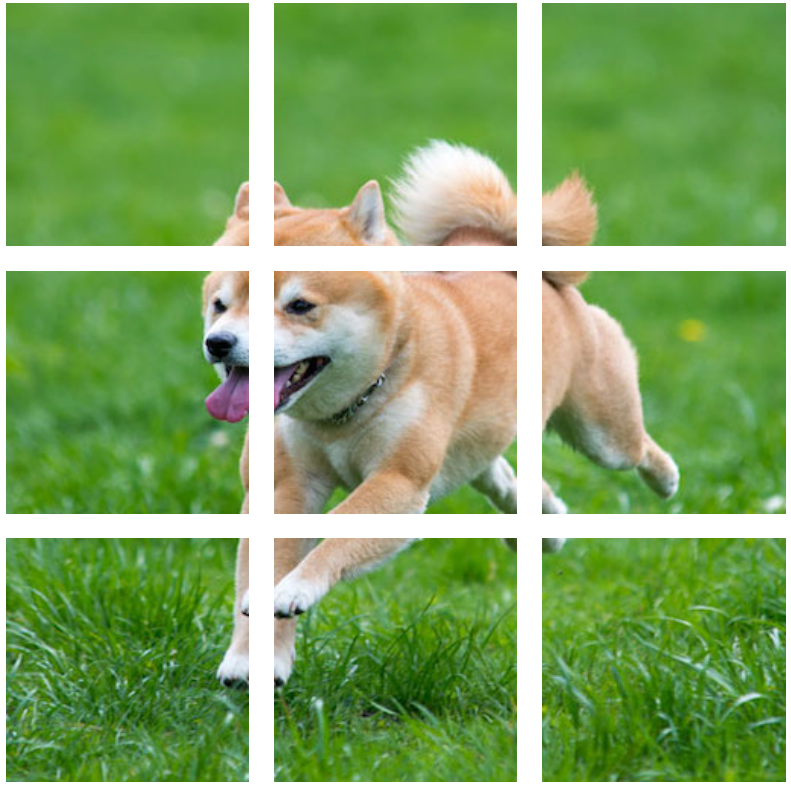


- Here, the patches do not overlap.
- The patches can overlap.
- User specifies:
  - patch size, e.g., 16×16;
  - stride, e.g., 16×16.

Slides courtesy of Dr. Shusen Wang

# Vectorization

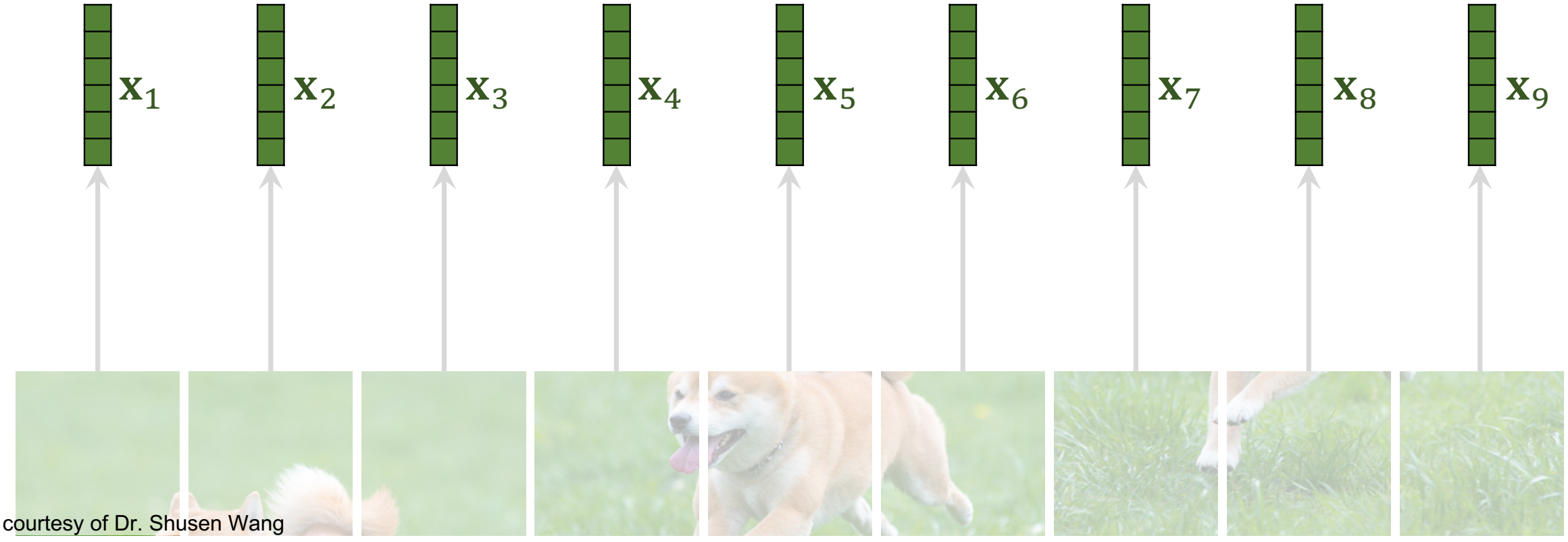
# Vectorization



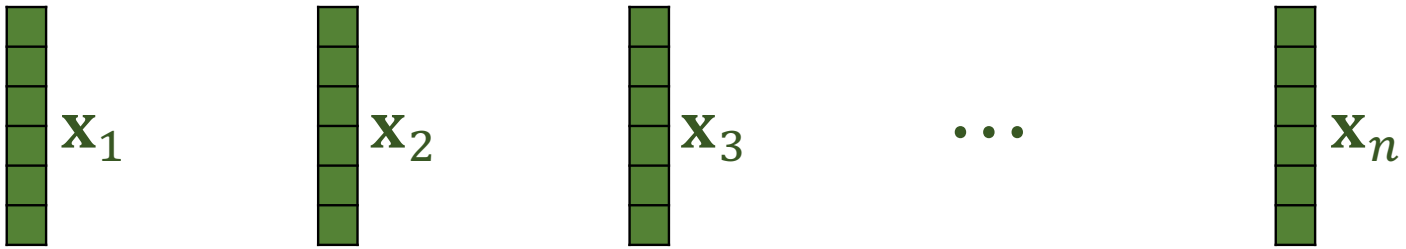
Slides courtesy of Dr. Shusen Wang

# Vectorization

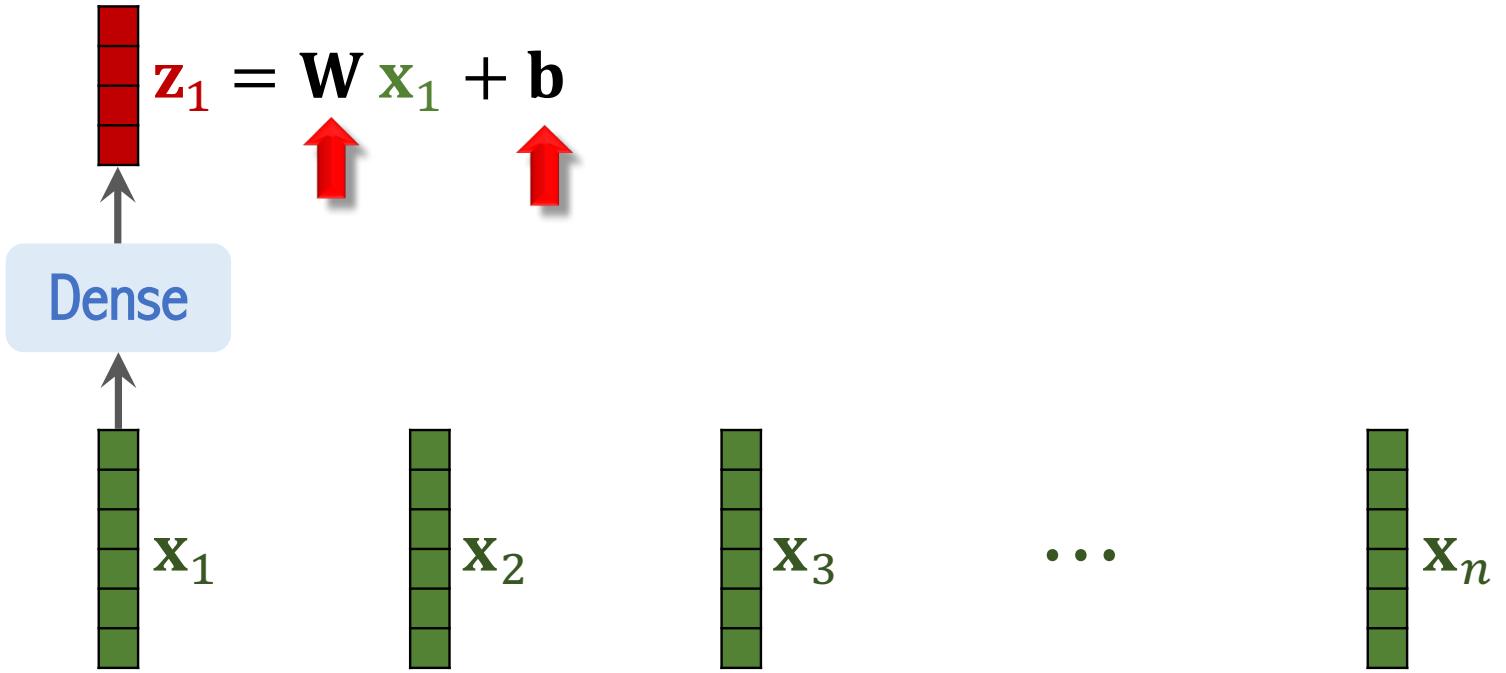
If the patches are $d_1 \times d_2 \times d_3$ tensors, then the vectors are $d_1 d_2 d_3 \times 1$.



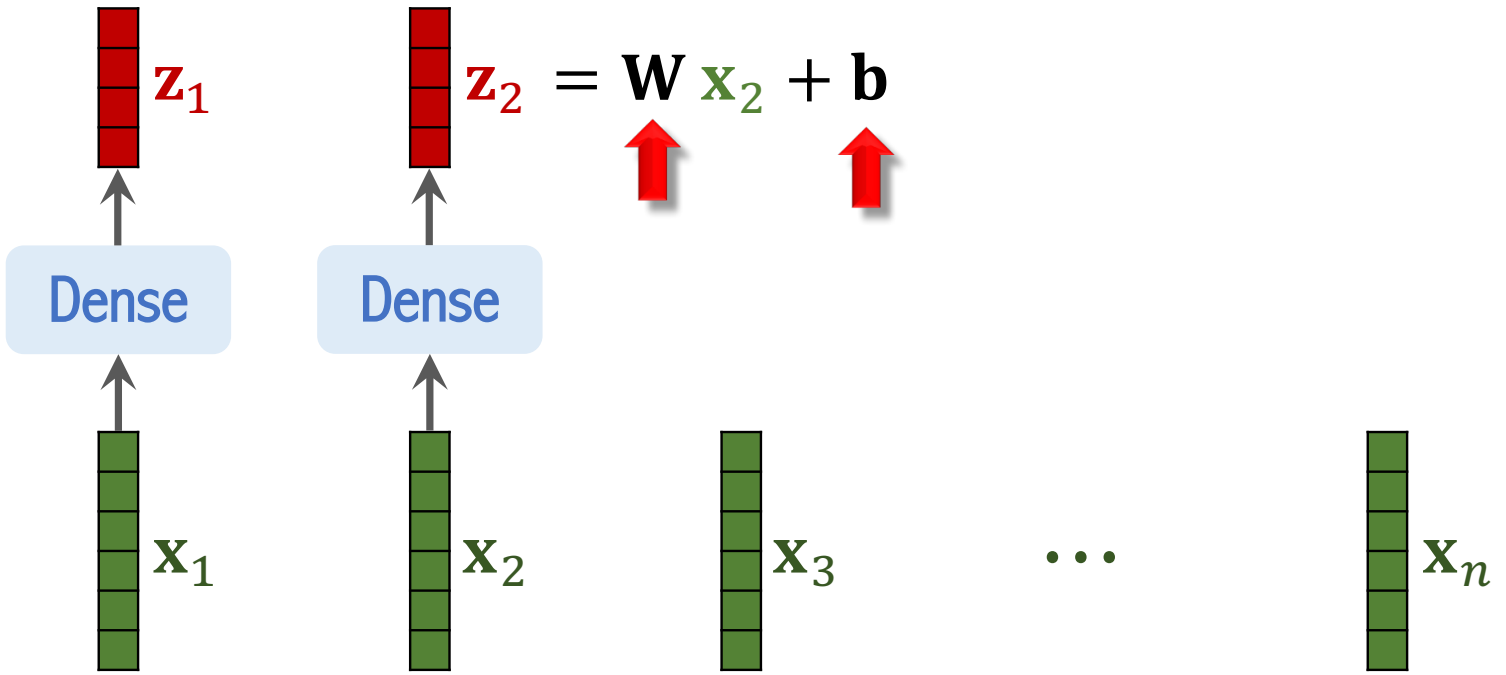

Slides courtesy of Dr. Shusen Wang



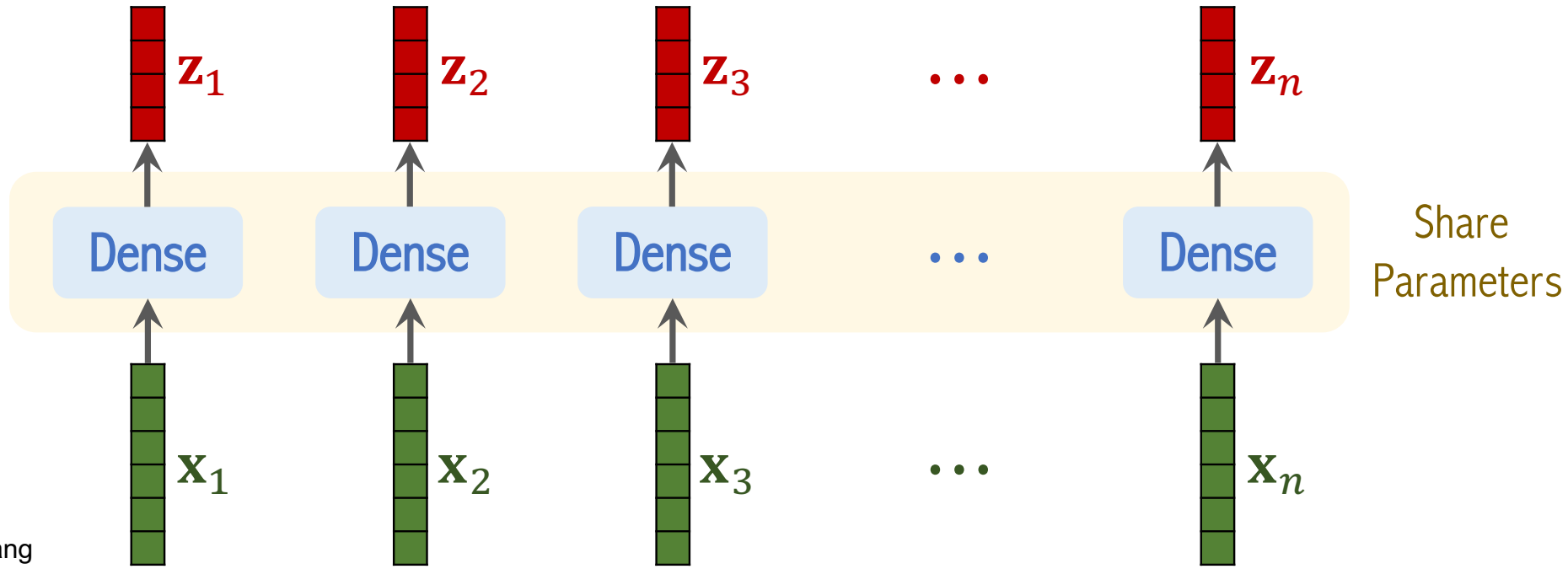

Slides courtesy of Dr. Shusen Wang




Slides courtesy of Dr. Shusen Wang



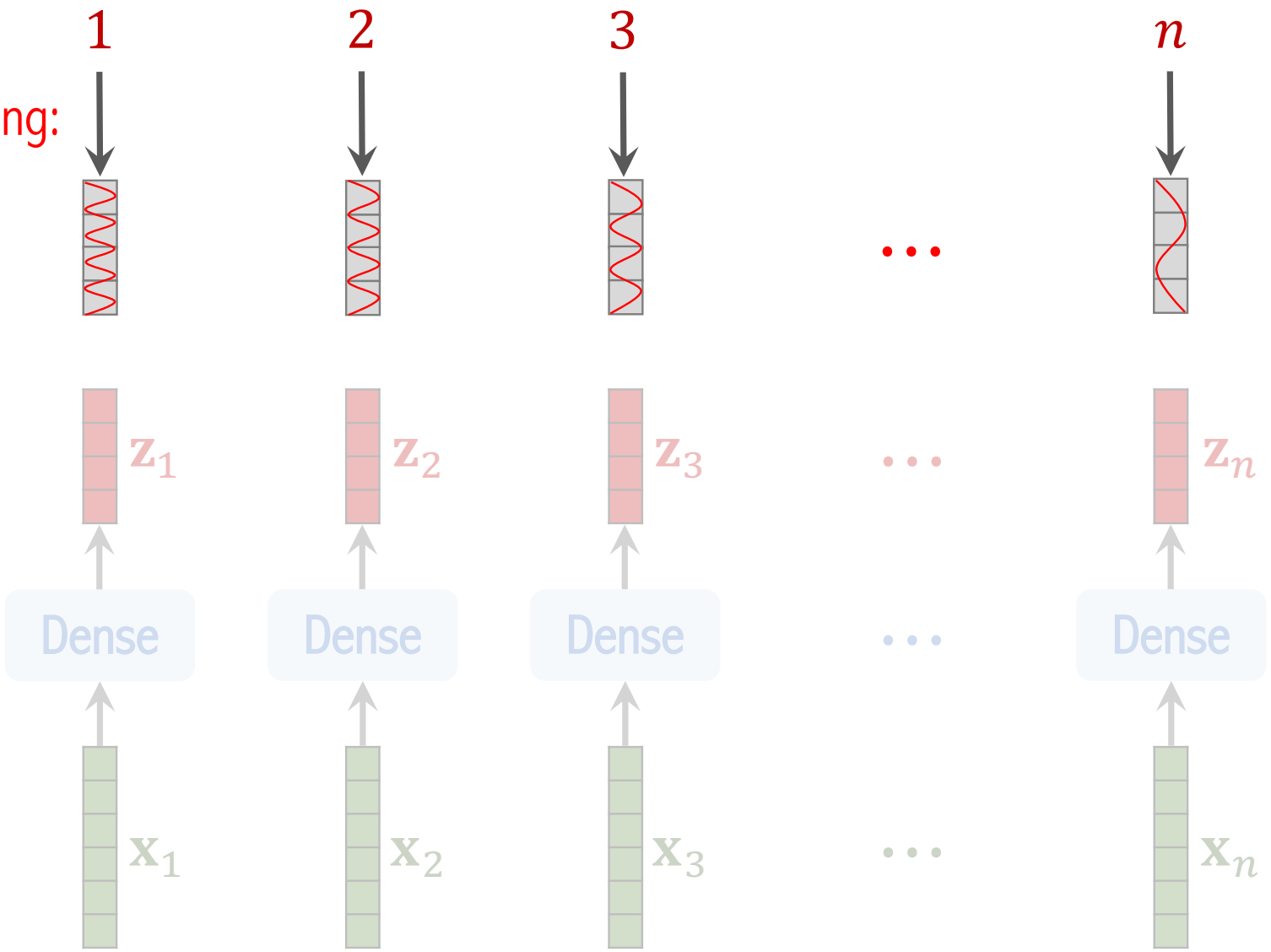

Slides courtesy of Dr. Shusen Wang




Slides courtesy of Dr. Shusen Wang

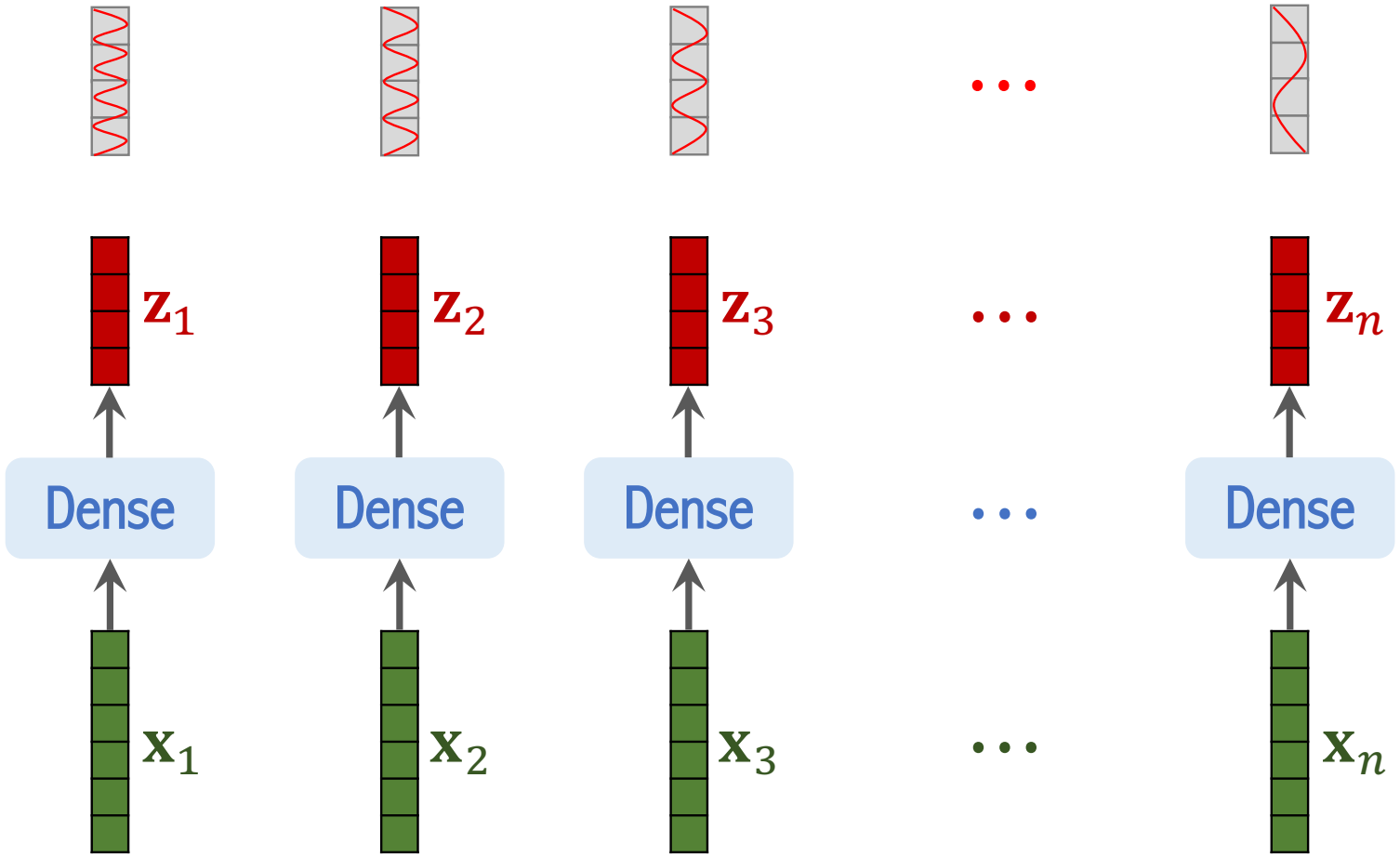
Interactive tool: https://erdem.pl/2021/05/understanding-positional-encoding-in-transformers

Positional Encoding:




Slides courtesy of Dr. Shusen Wang

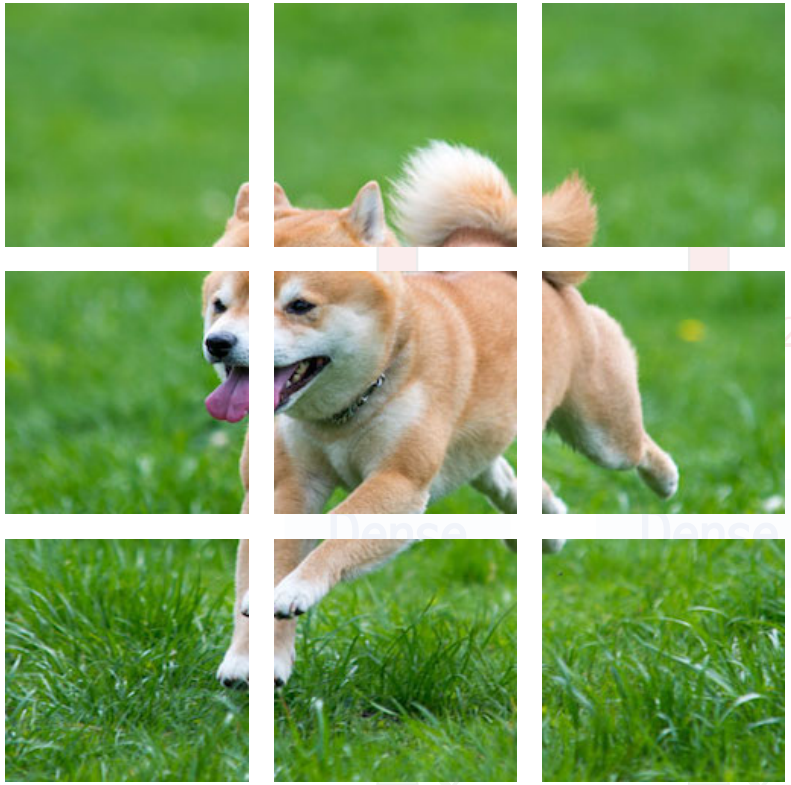
Add positional encoding vectors to $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$.



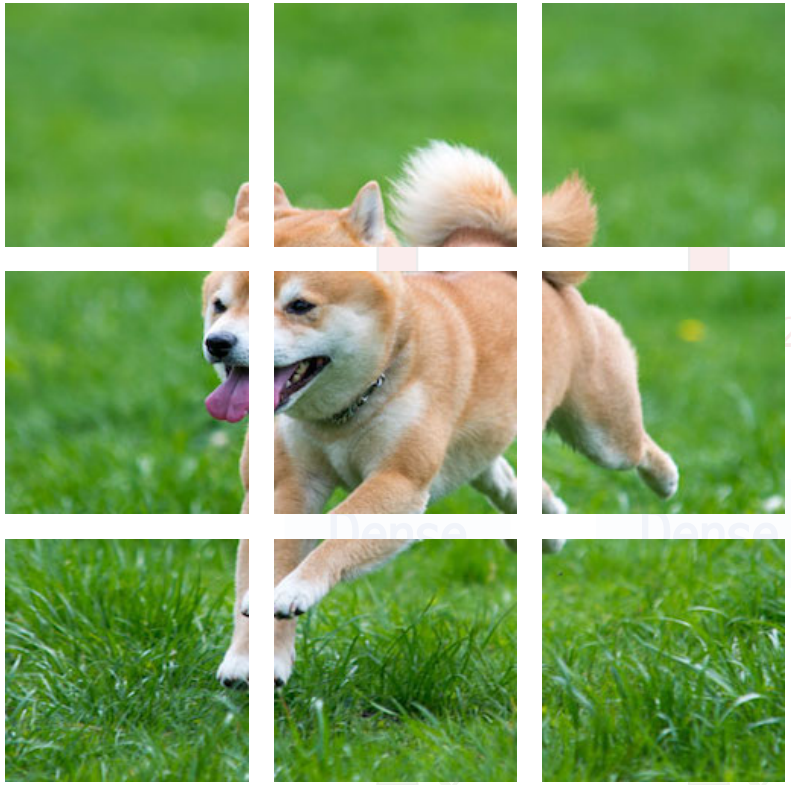

Slides courtesy of Dr. Shusen Wang

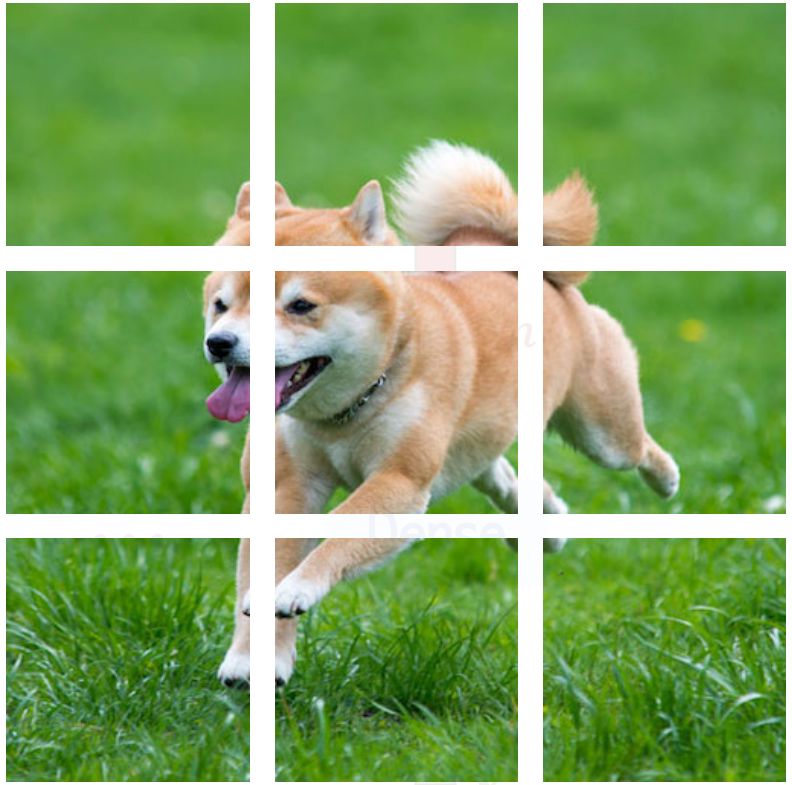
Add positional encoding vectors to $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$. (Why?)

Add positional encoding vectors to $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$. (Why?)



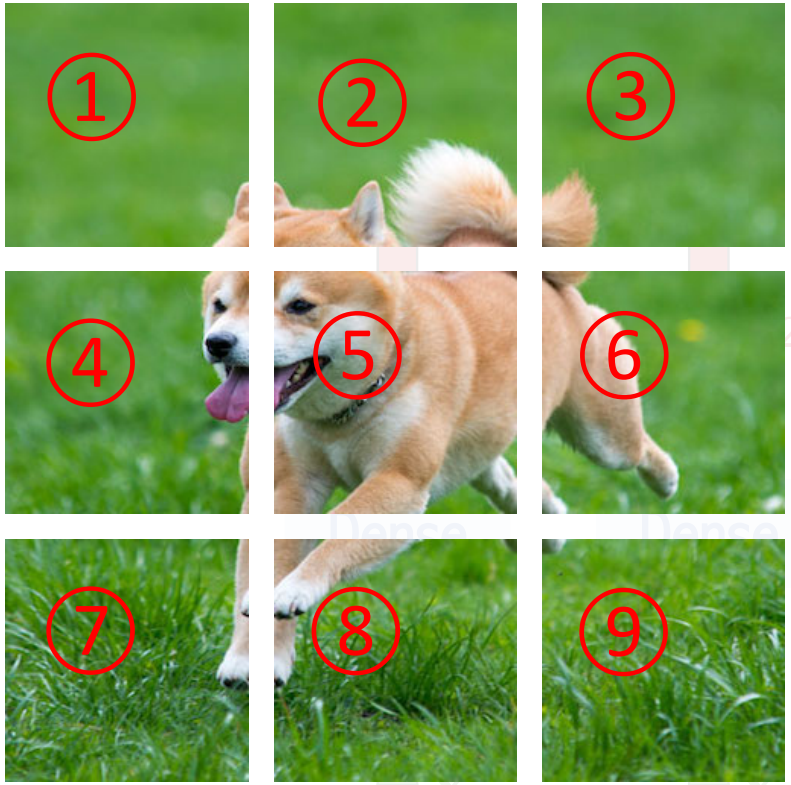
Add positional encoding vectors to $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$. (Why?)

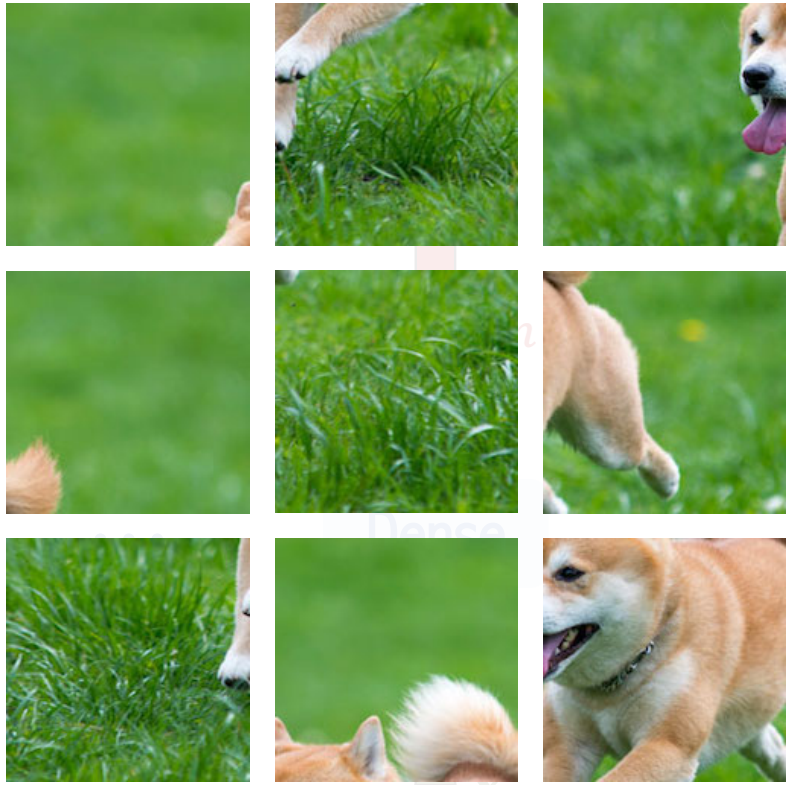




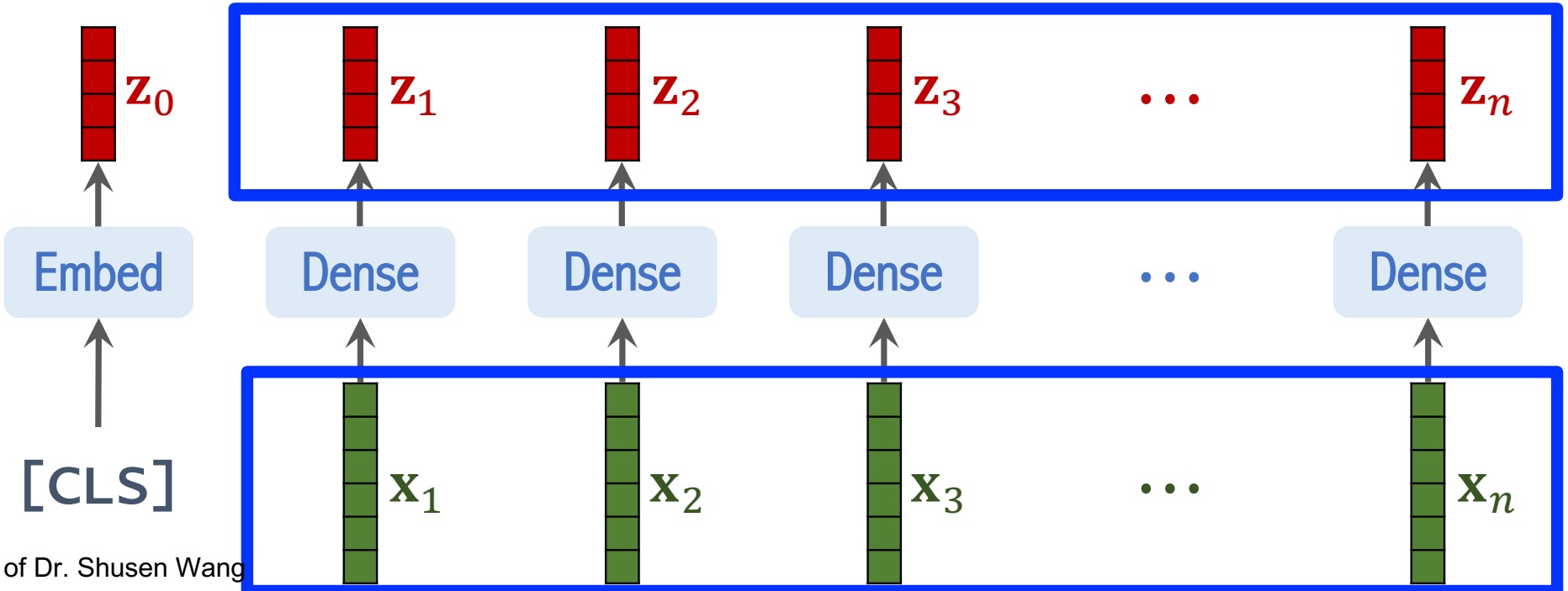
Slides courtesy of Dr. Shusen Wang

Add positional encoding vectors to $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$. (Why?)

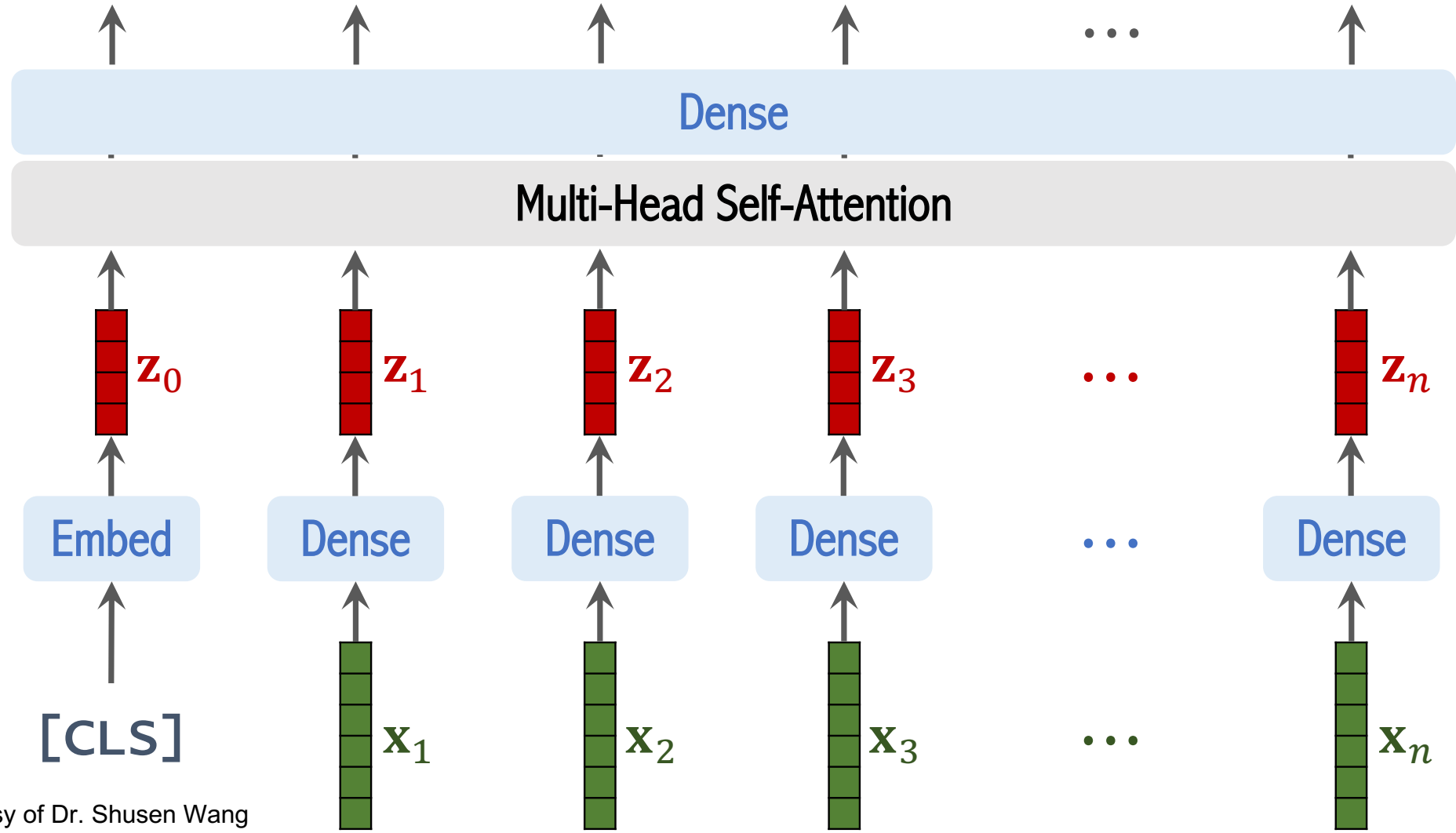


Slides courtesy of Dr. Shusen Wang




Slides courtesy of Dr. Shusen Wang

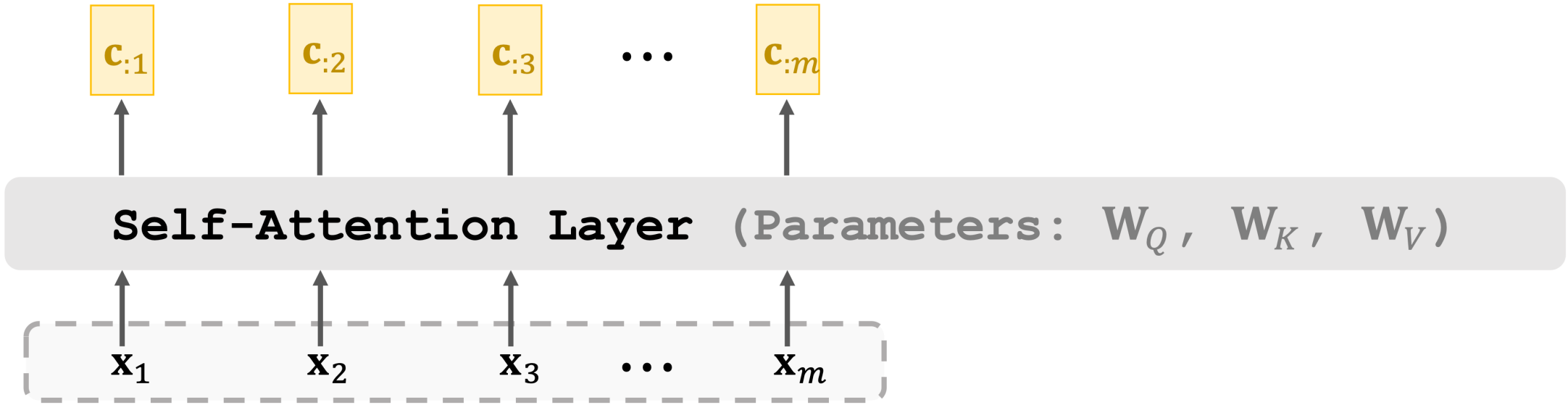



Slides courtesy of Dr. Shusen Wang

Detour

# Self-Attention Layer

- Self-attention layer: $\mathbf{C} = \mathrm{Attn}(\mathbf{X}, \mathbf{X})$.
  - Inputs: $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m]$.
  - Parameters: $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_V$.




Slides courtesy of Dr. Shusen Wang

Detour

# Self-Attention Layer

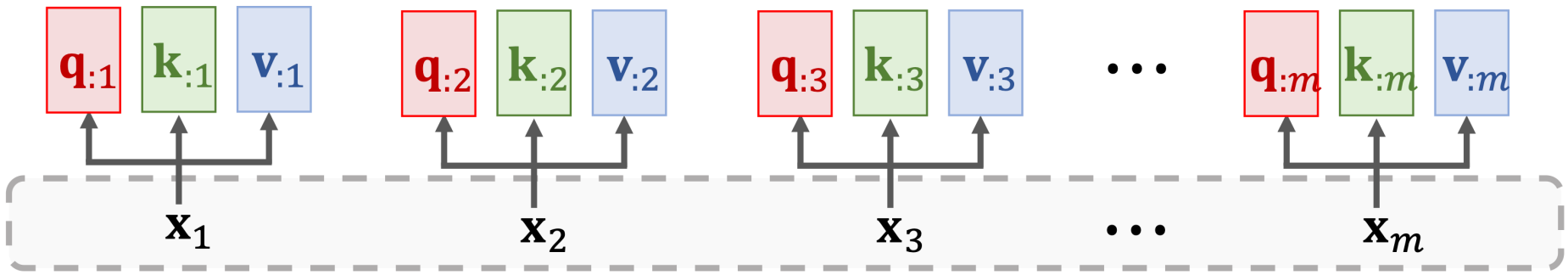
**Inputs:**

$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$ $\cdots$ $\mathbf{x}_m$

Detour

# Self-Attention Layer

Query: $\mathbf{q}_{:i} = \mathbf{W}_Q \mathbf{x}_i,$ Key: $\mathbf{k}_{:i} = \mathbf{W}_K \mathbf{x}_i,$ Value: $\mathbf{v}_{:i} = \mathbf{W}_V \mathbf{x}_i.$

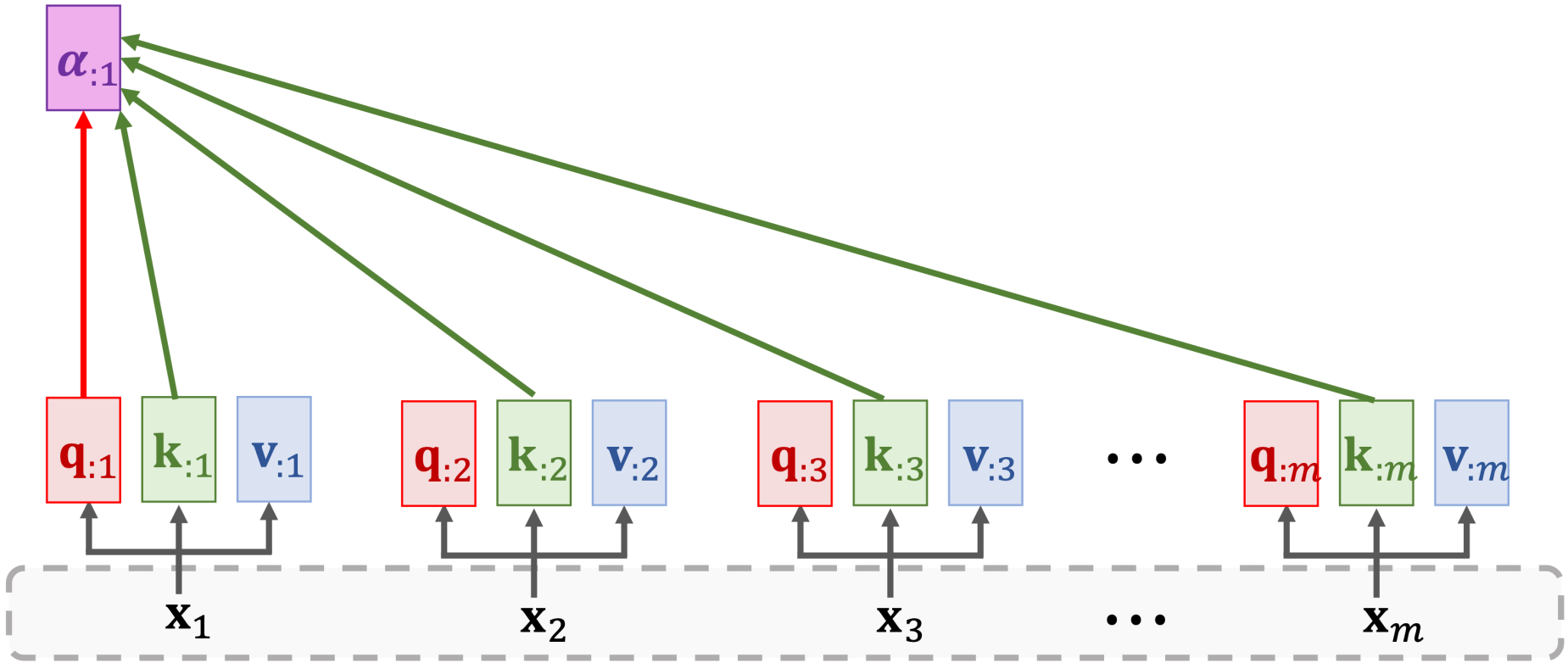



Slides courtesy of Dr. Shusen Wang

Detour

# Self-Attention Layer

**Weights:** $\alpha_{:j} = \text{Softmax}\left(\mathbf{K}^T \mathbf{q}_{:j}\right) \in \mathbb{R}^m.$

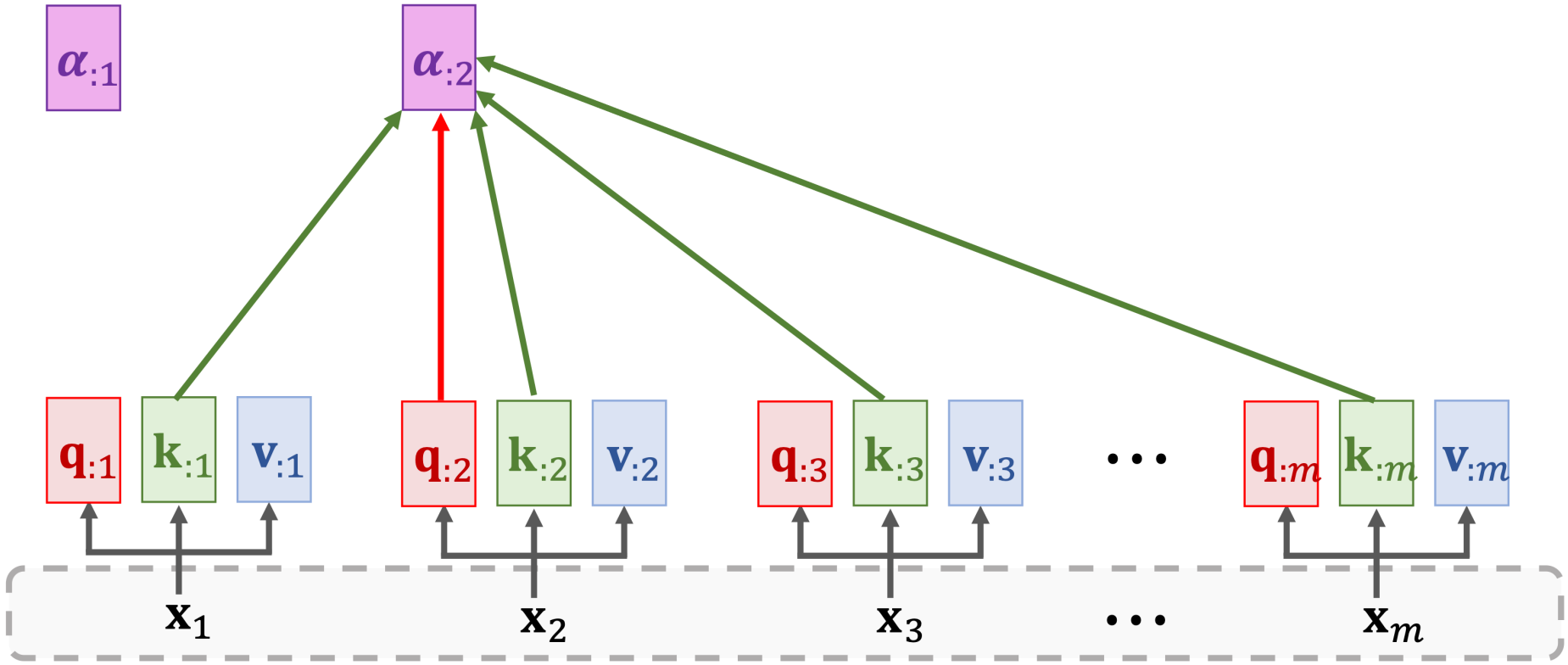



Slides courtesy of Dr. Shusen Wang

Detour

# Self-Attention Layer

**Weights:** $\alpha_{:j} = \text{Softmax}\left(\mathbf{K}^T \mathbf{q}_{:j}\right) \in \mathbb{R}^m.$

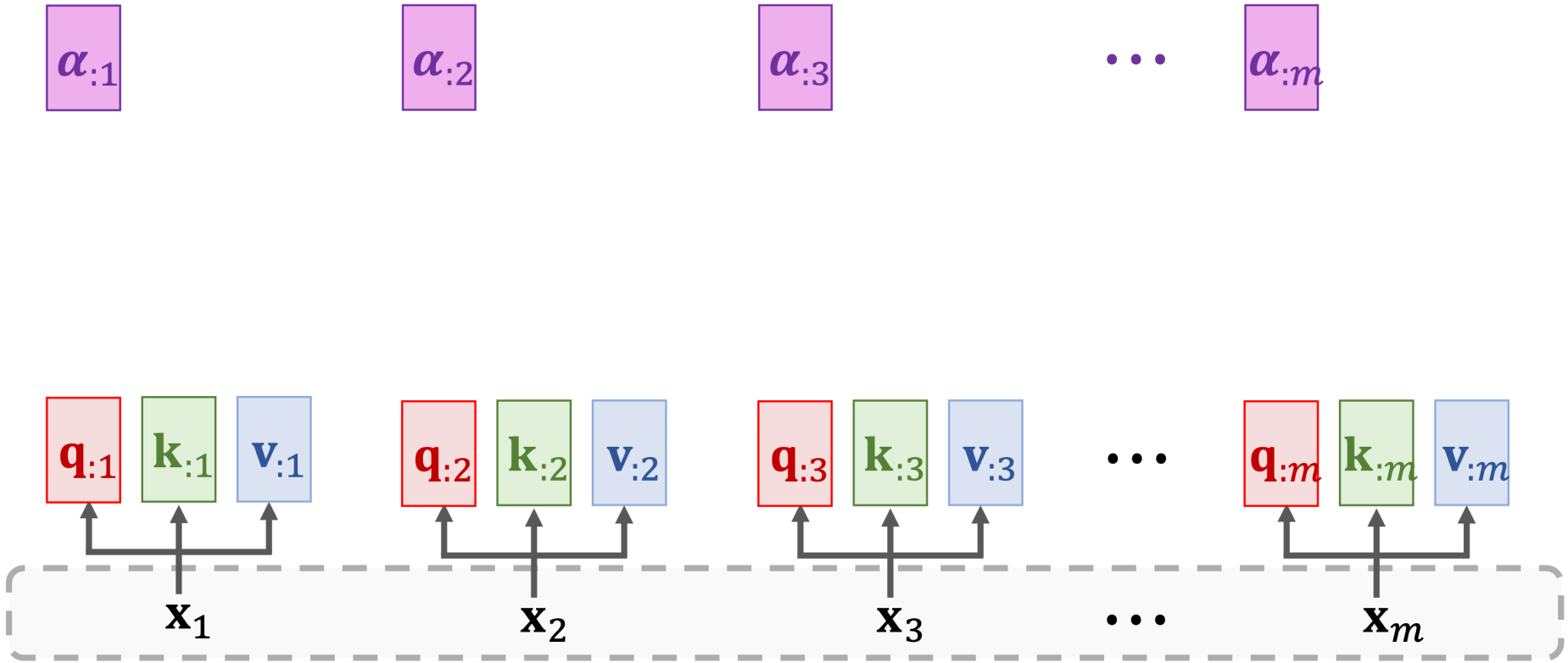



Slides courtesy of Dr. Shusen Wang

Detour

# Self-Attention Layer

**Weights:** $\alpha_{:j} = \text{Softmax}\left(\mathbf{K}^T \mathbf{q}_{:j}\right) \in \mathbb{R}^m.$

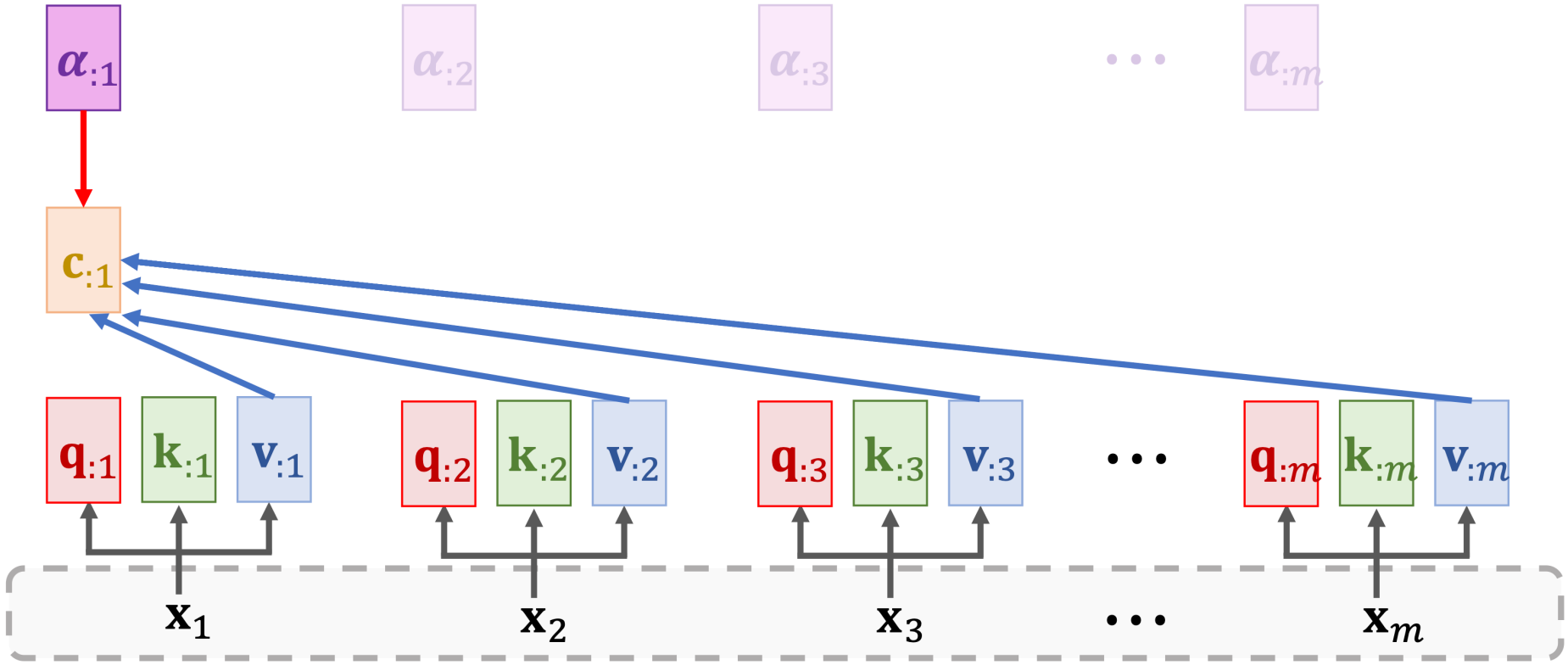



Slides courtesy of Dr. Shusen Wang

Detour

# Self-Attention Layer

**Context vector:** $\mathbf{c}_{:1} = \alpha_{11}\mathbf{v}_{:1} + \cdots + \alpha_{m1}\mathbf{v}_{:m} = \mathbf{V}\boldsymbol{\alpha}_{:1}$.

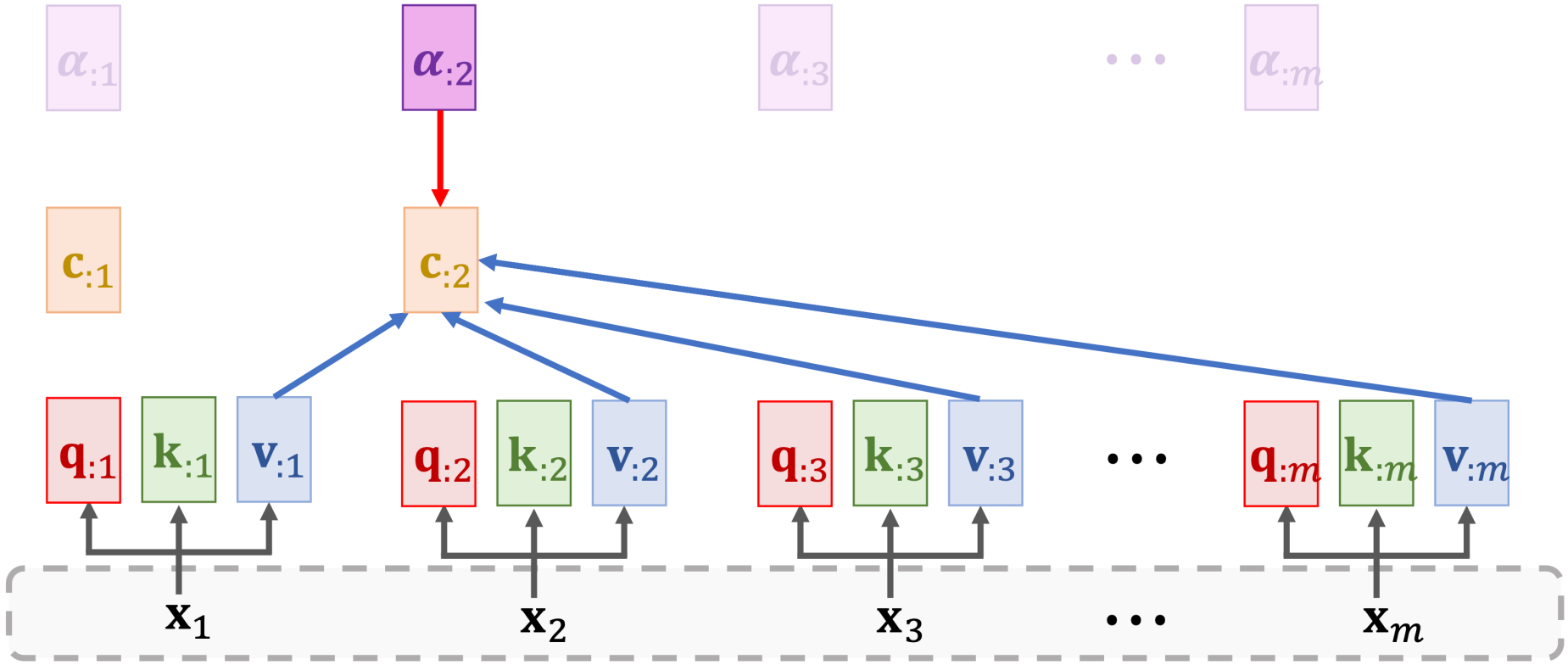



Slides courtesy of Dr. Shusen Wang

Detour

# Self-Attention Layer

**Context vector:** $\mathbf{c}_{:2} = \alpha_{12}\mathbf{v}_{:1} + \cdots + \alpha_{m2}\mathbf{v}_{:m} = \mathbf{V}\boldsymbol{\alpha}_{:2}.$

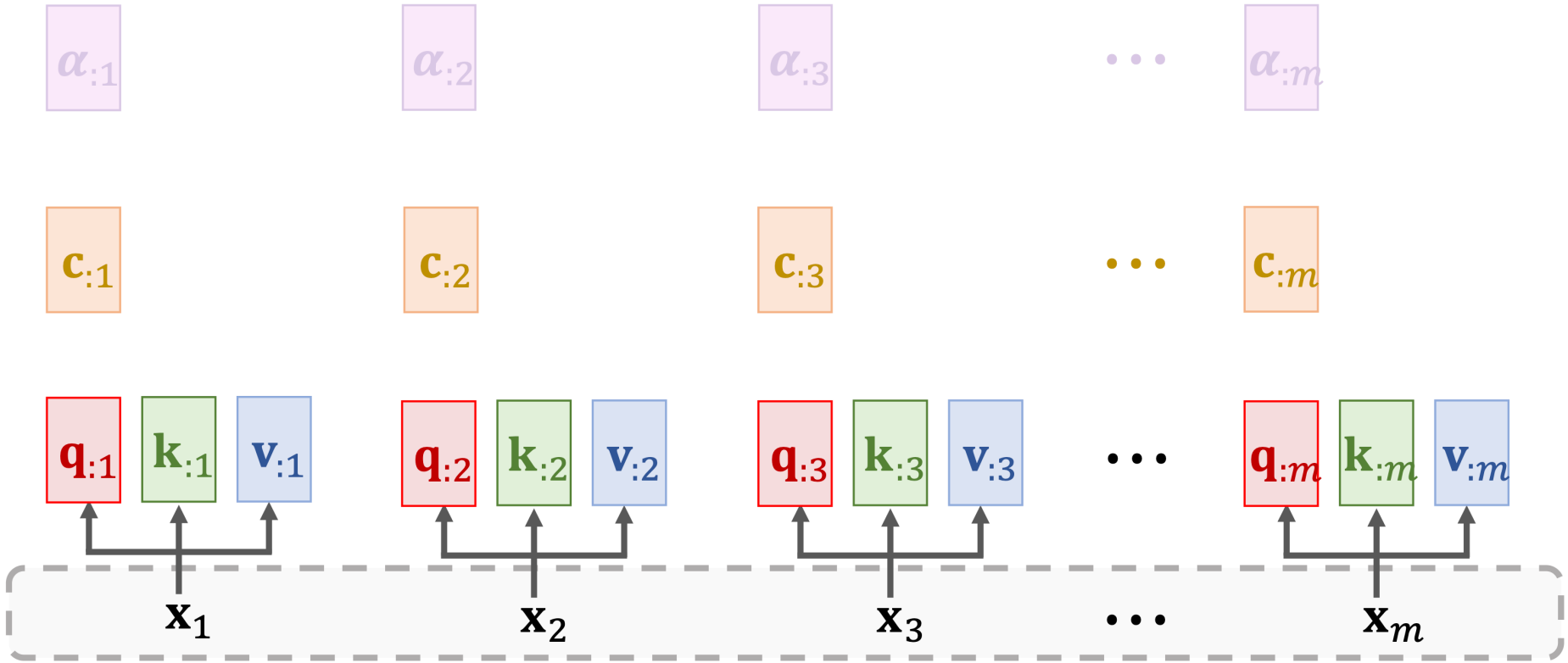



Slides courtesy of Dr. Shusen Wang

Detour

# Self-Attention Layer

**Context vector:** $\mathbf{c}_{:j} = \alpha_{1j}\mathbf{v}_{:1} + \cdots + \alpha_{mj}\mathbf{v}_{:m} = \mathbf{V}\boldsymbol{\alpha}_{:j}$.



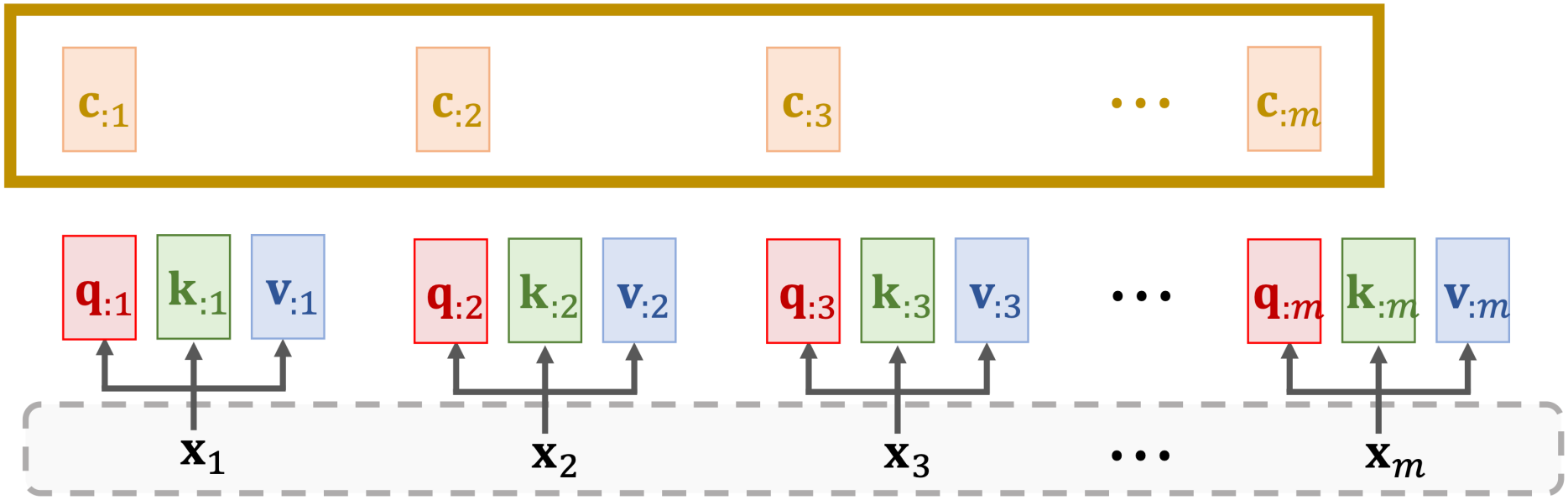

Slides courtesy of Dr. Shusen Wang

Detour

# Self-Attention Layer

- Here, $\mathbf{c}_{:j} = \mathbf{V} \cdot \text{Softmax}\left(\mathbf{K}^T \mathbf{q}_{:j}\right)$.
- Thus, $\mathbf{c}_{:j}$ is a function of all the $m$ vectors $\mathbf{x}_1, \cdots, \mathbf{x}_m$.

**Output of self-attention layer:**

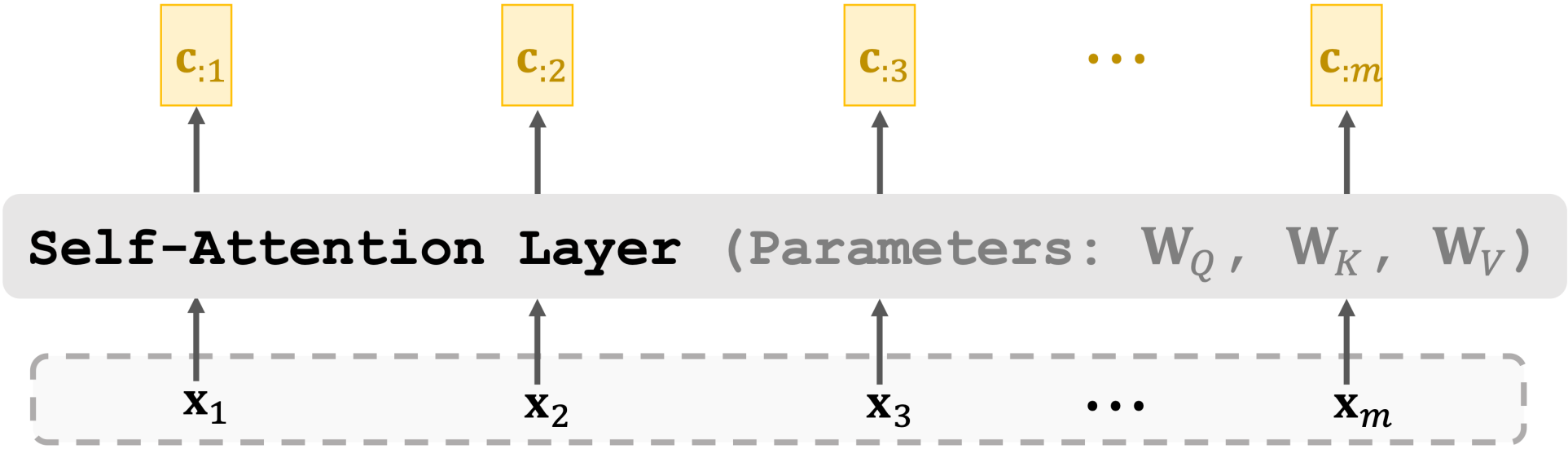



Slides courtesy of Dr. Shusen Wang

Detour

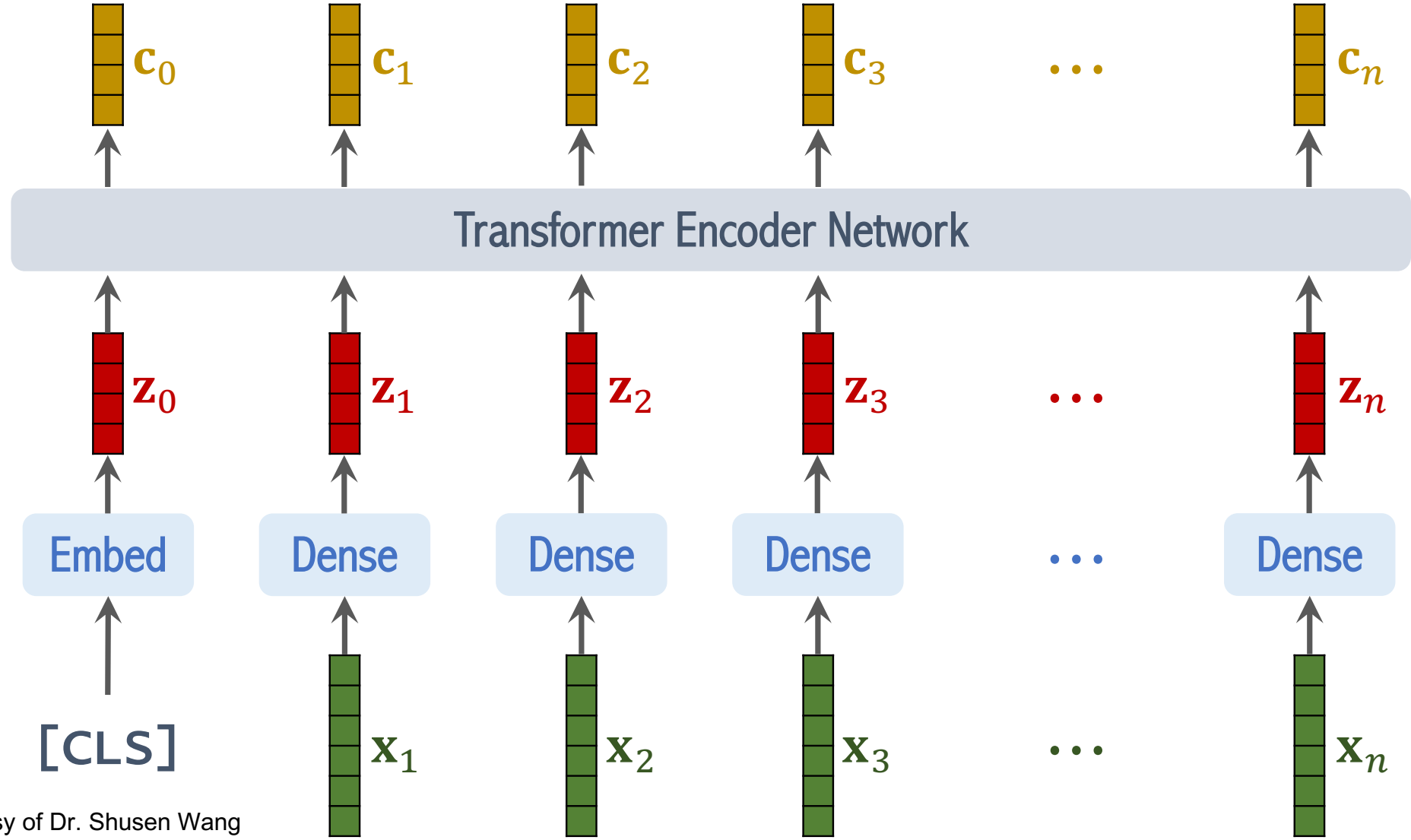
# Self-Attention Layer

- Self-attention layer: $\mathbf{C} = \mathrm{Attn}(\mathbf{X}, \mathbf{X})$.
  - Inputs: $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m]$.
  - Parameters: $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_V$.

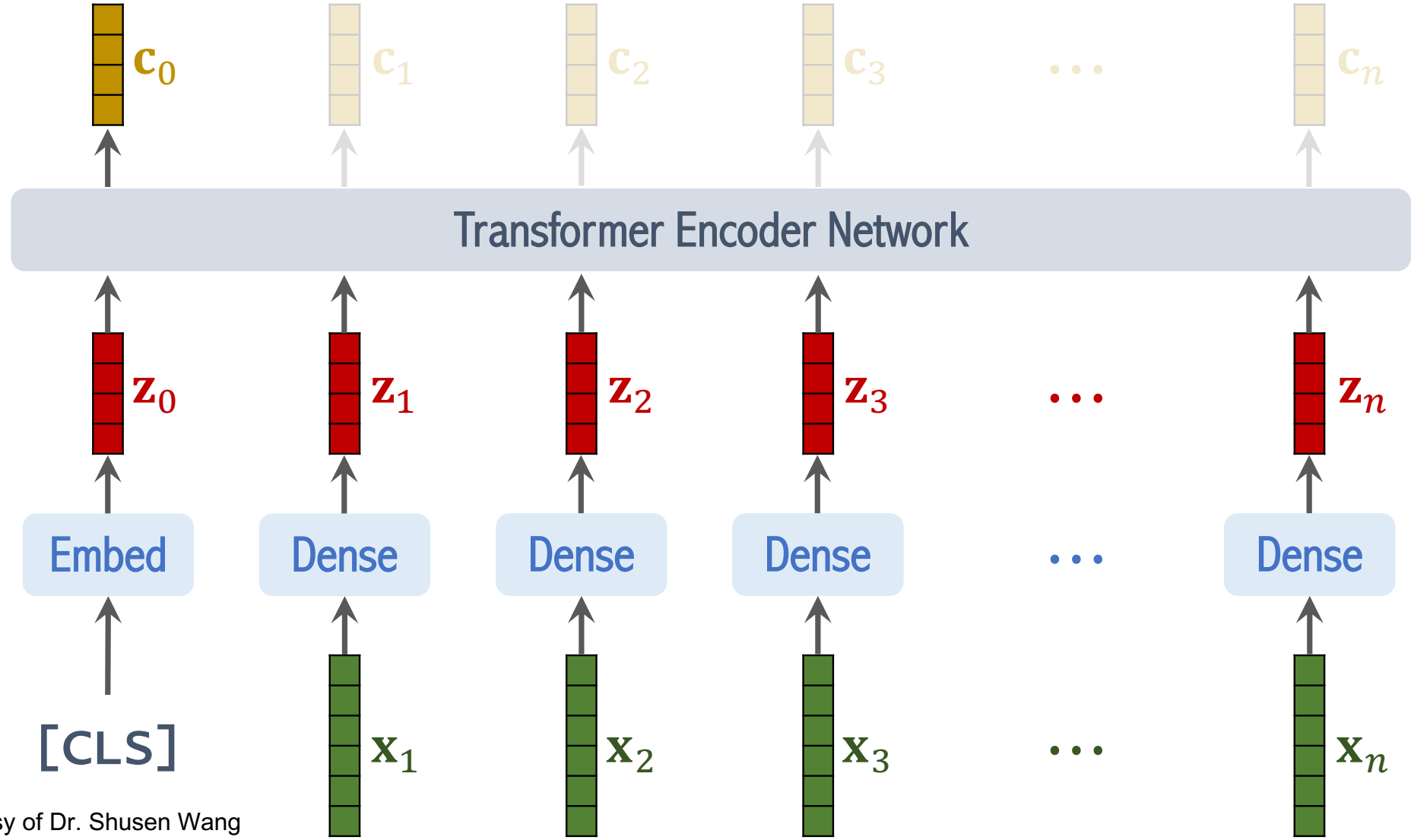



Slides courtesy of Dr. Shusen Wang

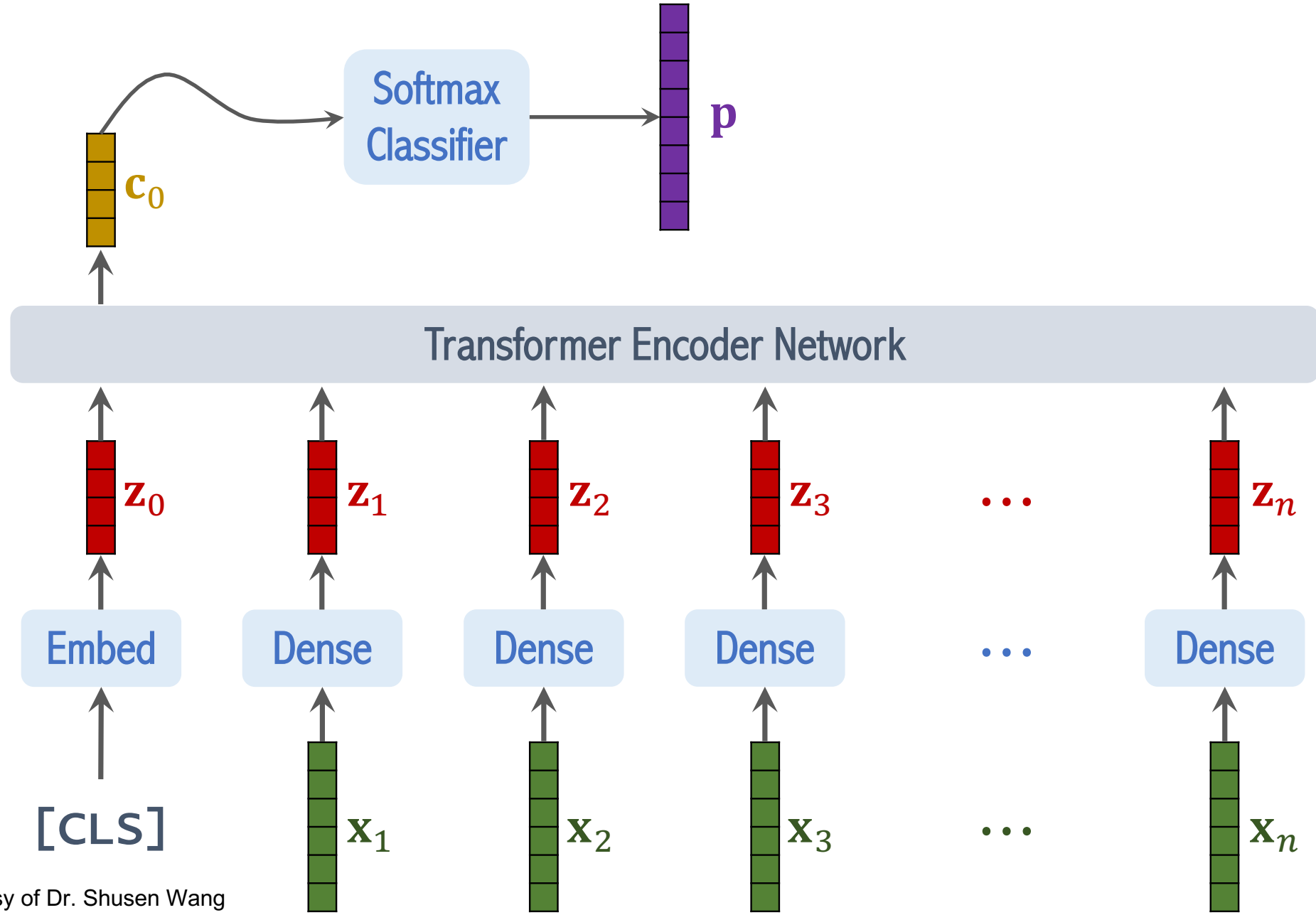

Slides courtesy of Dr. Shusen Wang



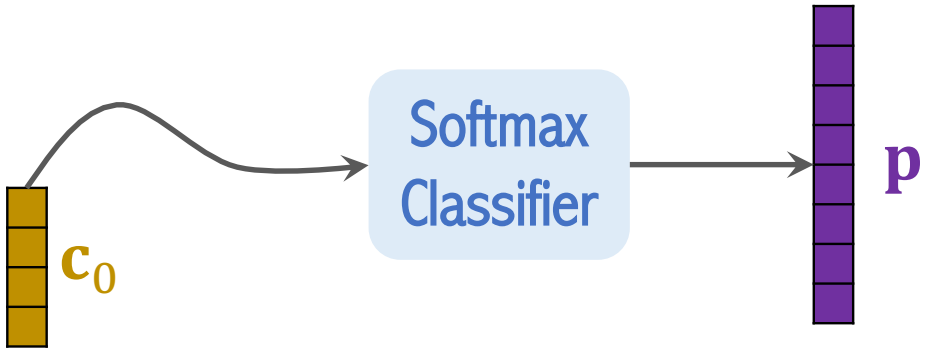

Slides courtesy of Dr. Shusen Wang

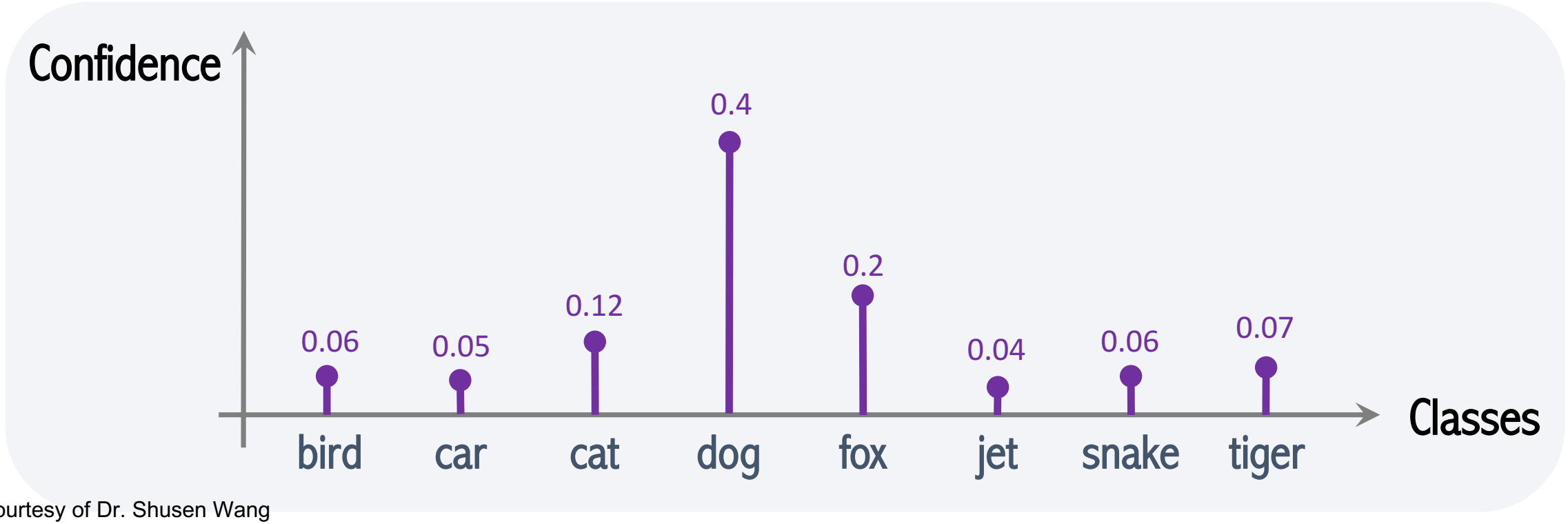



Slides courtesy of Dr. Shusen Wang

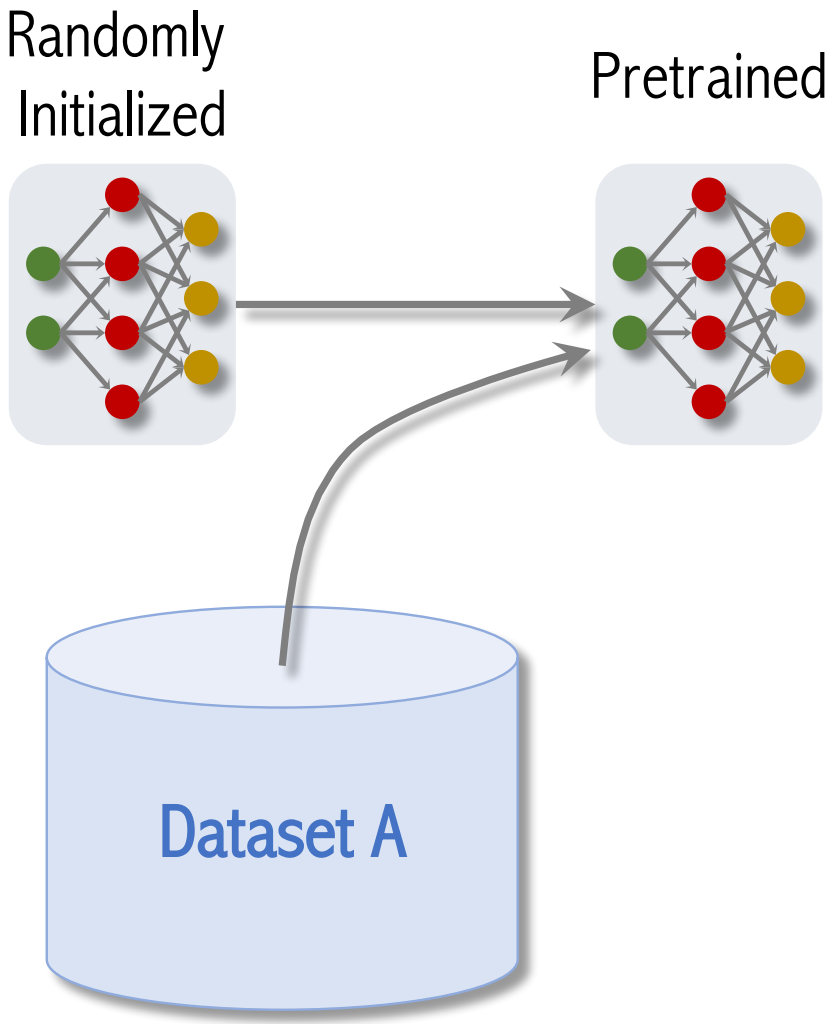



Slides courtesy of Dr. Shusen Wang

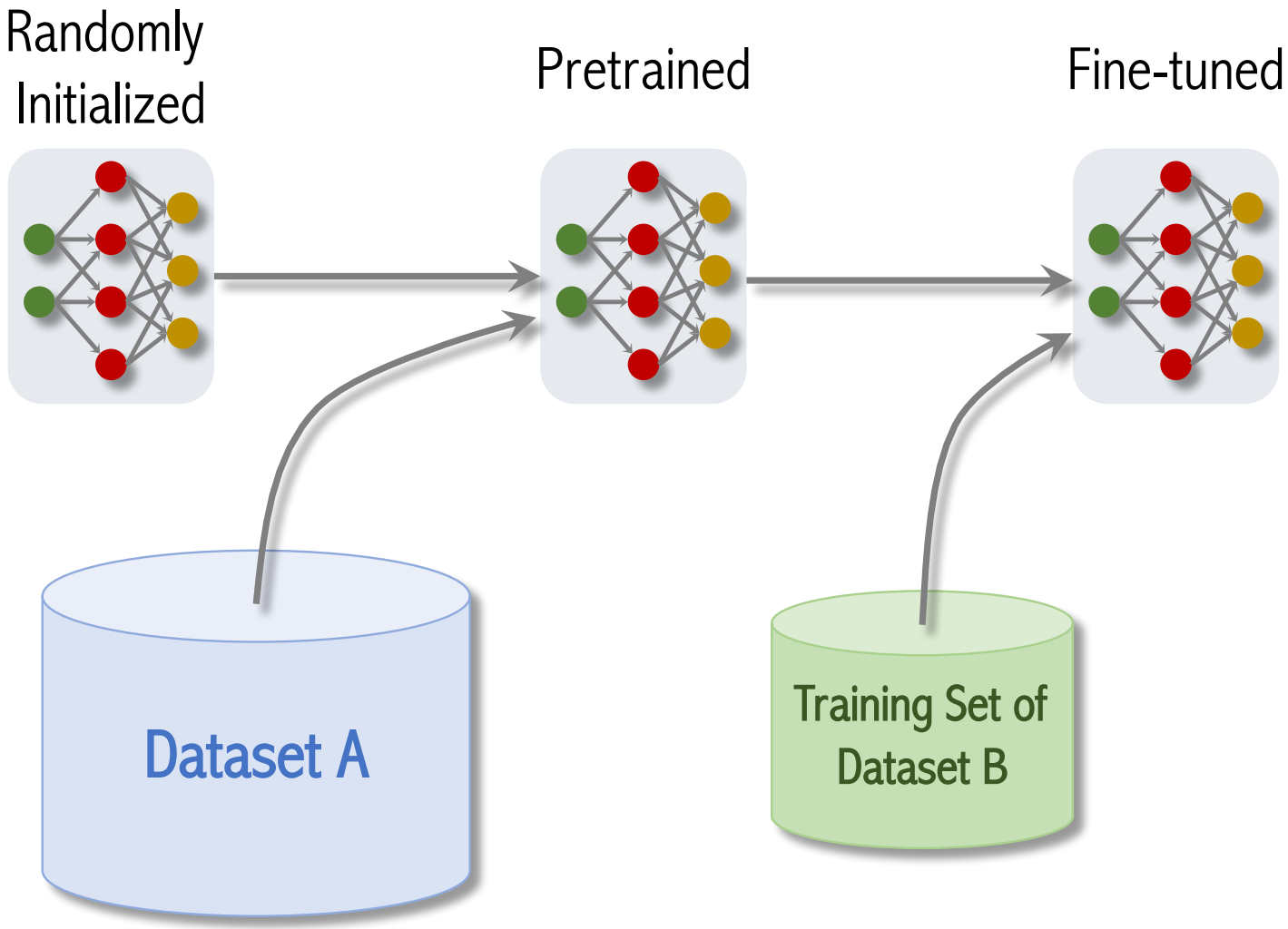


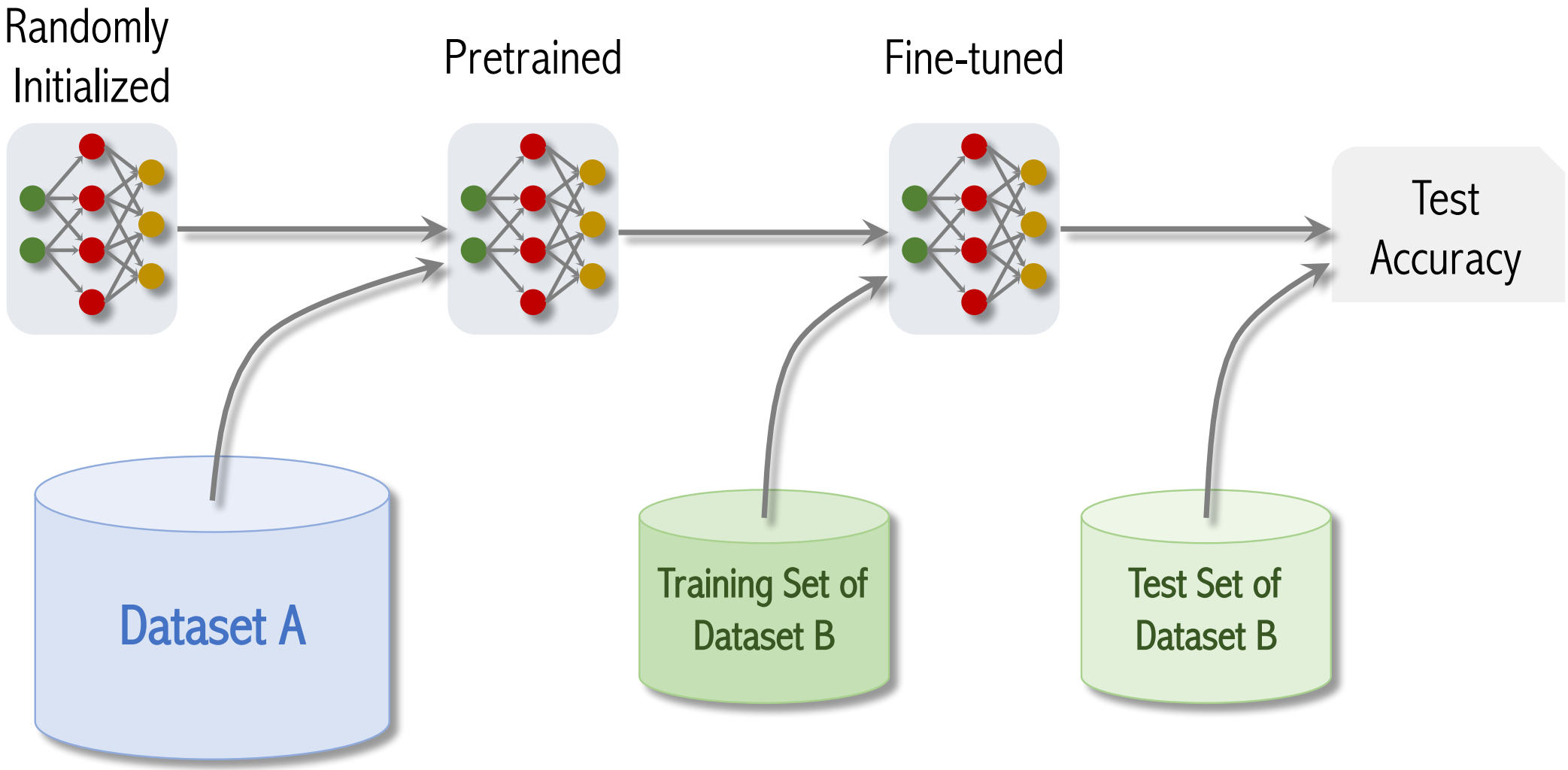

Slides courtesy of Dr. Shusen Wang




Slides courtesy of Dr. Shusen Wang




Slides courtesy of Dr. Shusen Wang




Slides courtesy of Dr. Shusen Wang

# Datasets

| | # of Images | # of Classes |
|---|---|---|
| ImageNet (Small) | 1.3 Million | 1 Thousand |
| ImageNet-21K (Medium) | 14 Million | 21 Thousand |
| JFT (Big) | 300 Million | 18 Thousand |

Slides courtesy of Dr. Shusen Wang

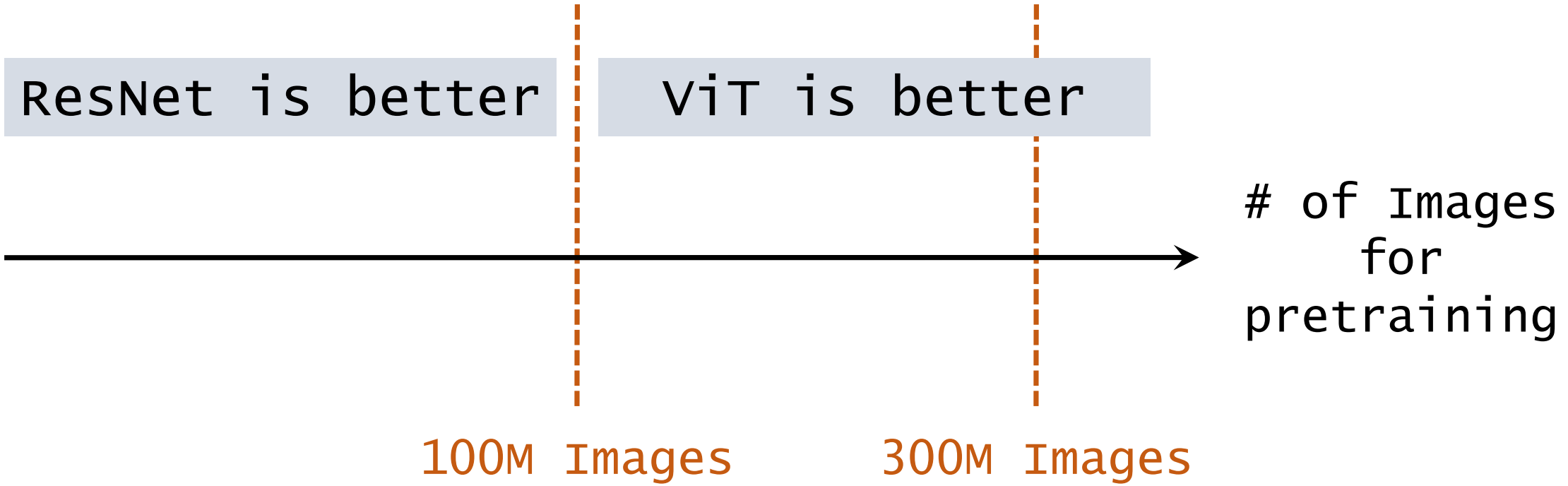
# Image Classification Accuracies

- Pretrain the model on Dataset A, fine-tune the model on Dataset B, and evaluate the model on Dataset B.

- Pretrained on ImageNet (small), ViT is slightly worse than ResNet.
- Pretrained on ImageNet-21K (medium), ViT is comparable to ResNet.
- Pretrained on JFT (large), ViT is slightly better than ResNet.

Slides courtesy of Dr. Shusen Wang

# Image Classification Accuracies




Slides courtesy of Dr. Shusen Wang

# Next

- Objection detection before deep learning
  - + AdaBoost
  - + HOG
- Object detection using deep learning
  - ++ R-CNN / Fast R-CNN / Faster R-CNN
  - + Anchor-based Region Proposal
  - + Single-stage Detector
  - + Anchor-free Detector

- References
  - SZ Chapter 6.3
  - Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., & Pietikäinen, M. (2020). Deep learning for generic object detection: A survey. International journal of computer vision, 128(2), 261-318.
  - Zou, Zhengxia, Zhenwei Shi, Yuhong Guo, and Jieping Ye. "Object detection in 20 years: A survey." arXiv preprint arXiv:1905.05055 (2019).