Parts and Pieces for deep NN

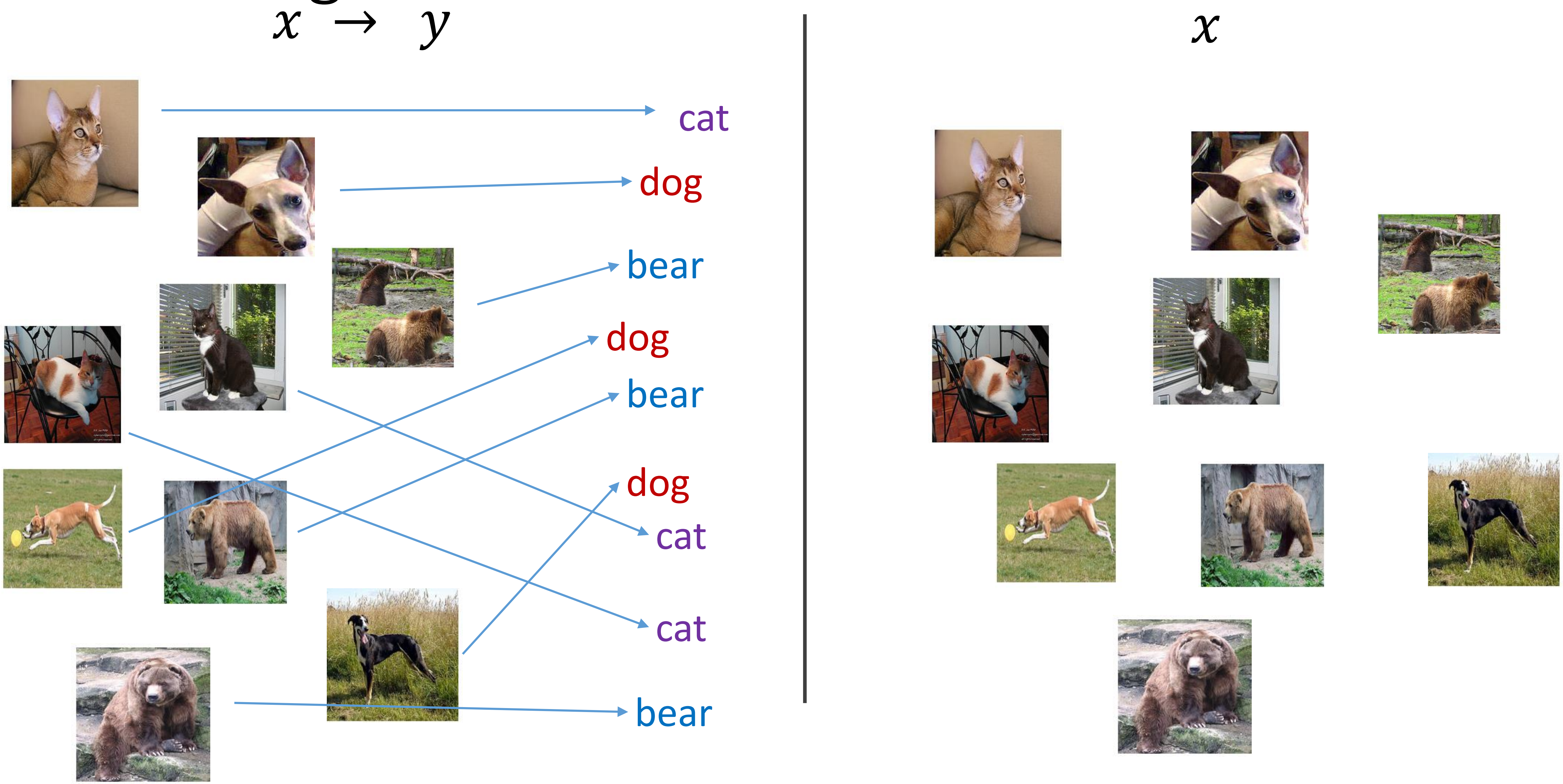
Biplab Banerjee

Summary: Image Features

- The idea of low, mid, and high level features
- Largely replaced by Neural networks
- But there is a direct connection between the feature hierarchy

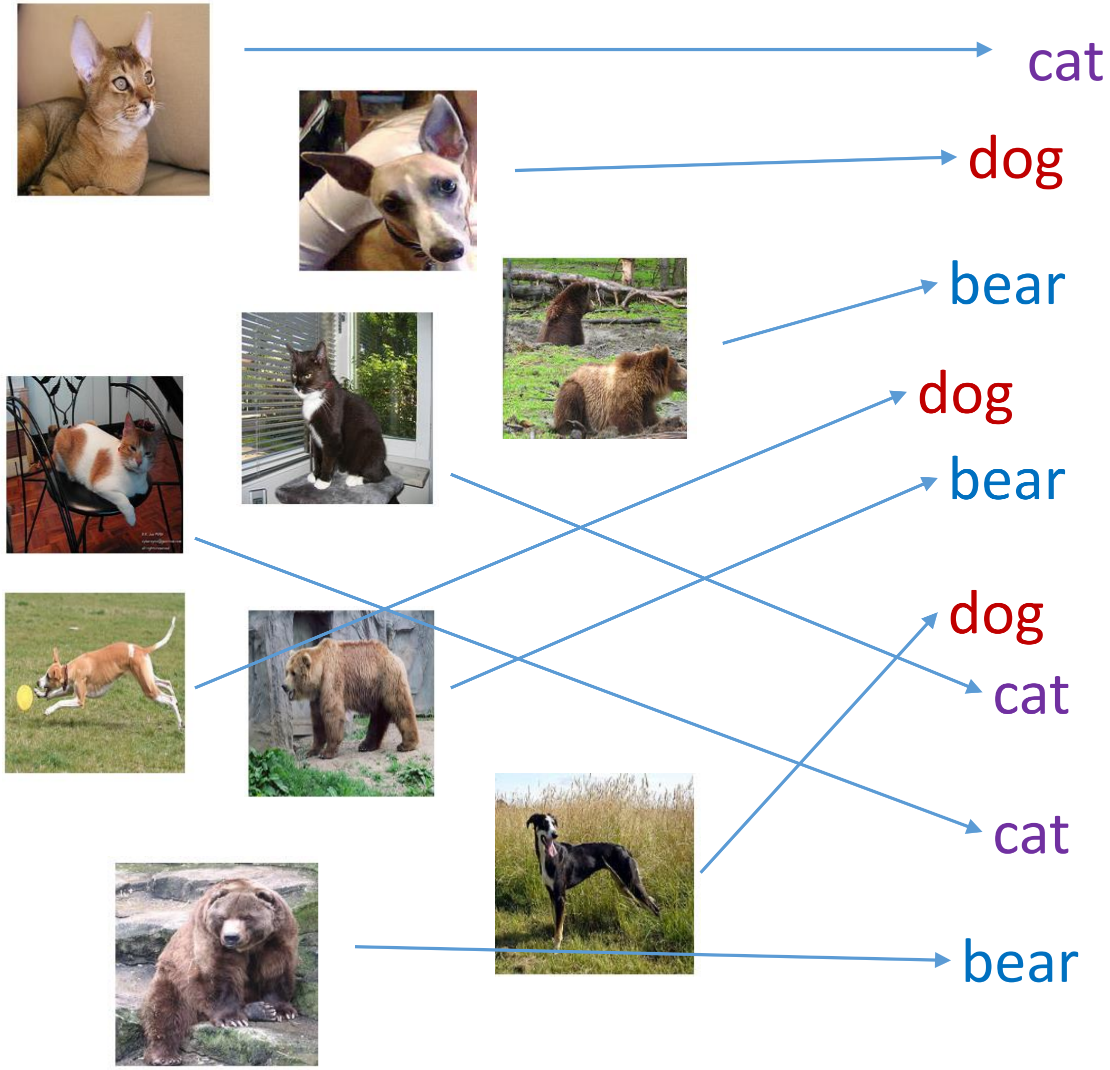
- Many other features proposed
  - LBP: Local Binary Patterns: Useful for recognizing faces.
  - Dense SIFT: SIFT features computed on a grid similar to the HOG features.
  - etc.

Supervised Learning vs Unsupervised Learning

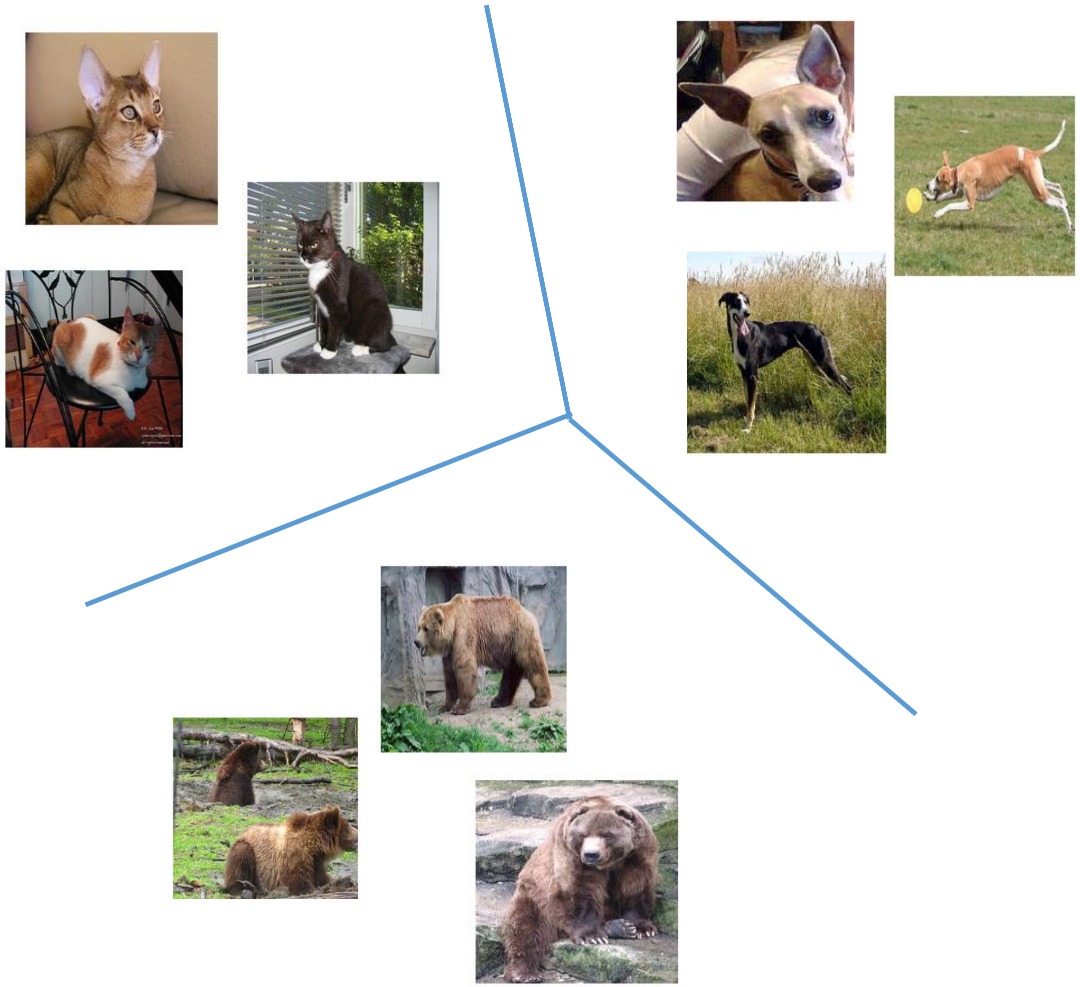


Supervised Learning vs Unsupervised Learning

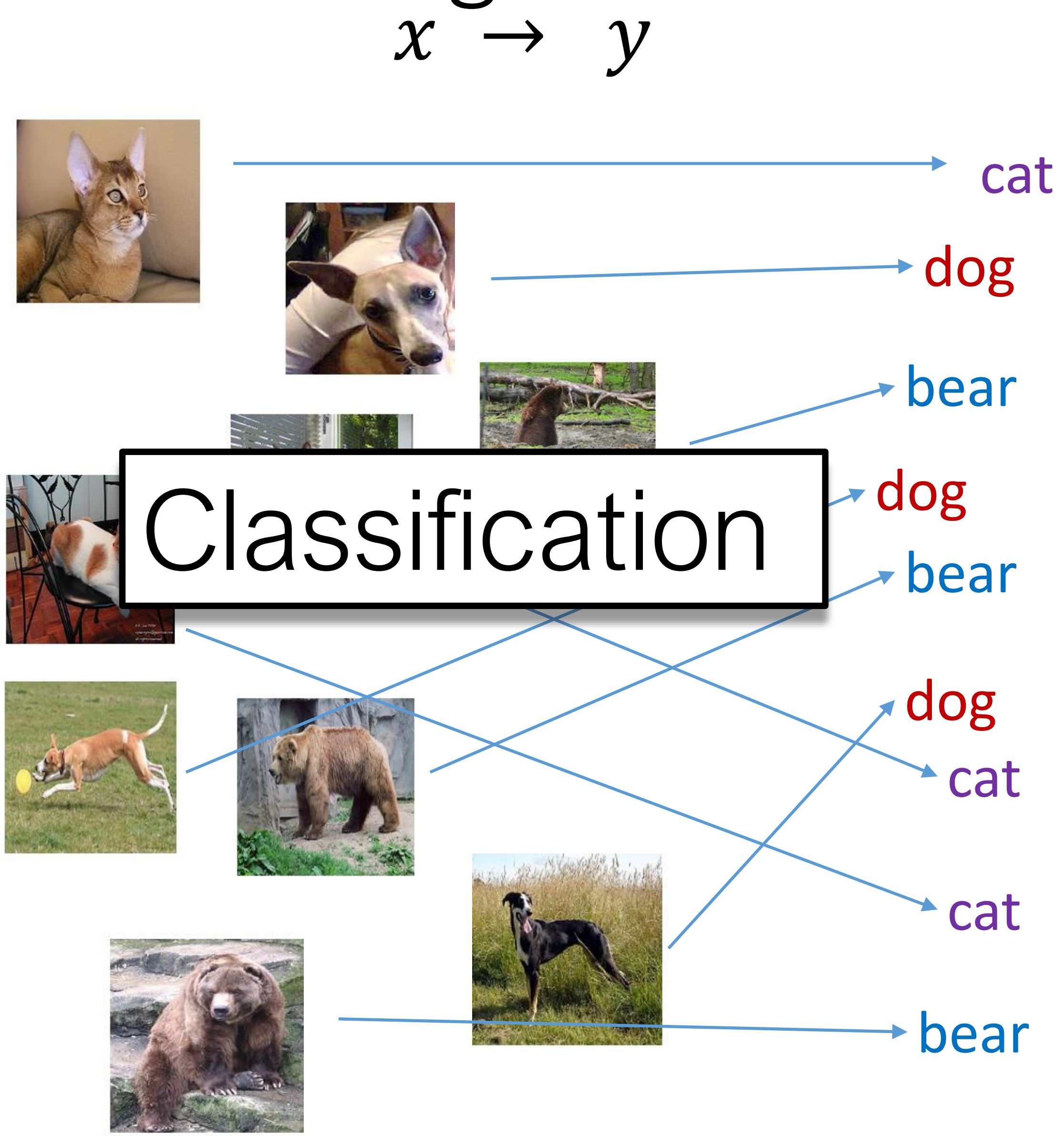
$x \rightarrow y$



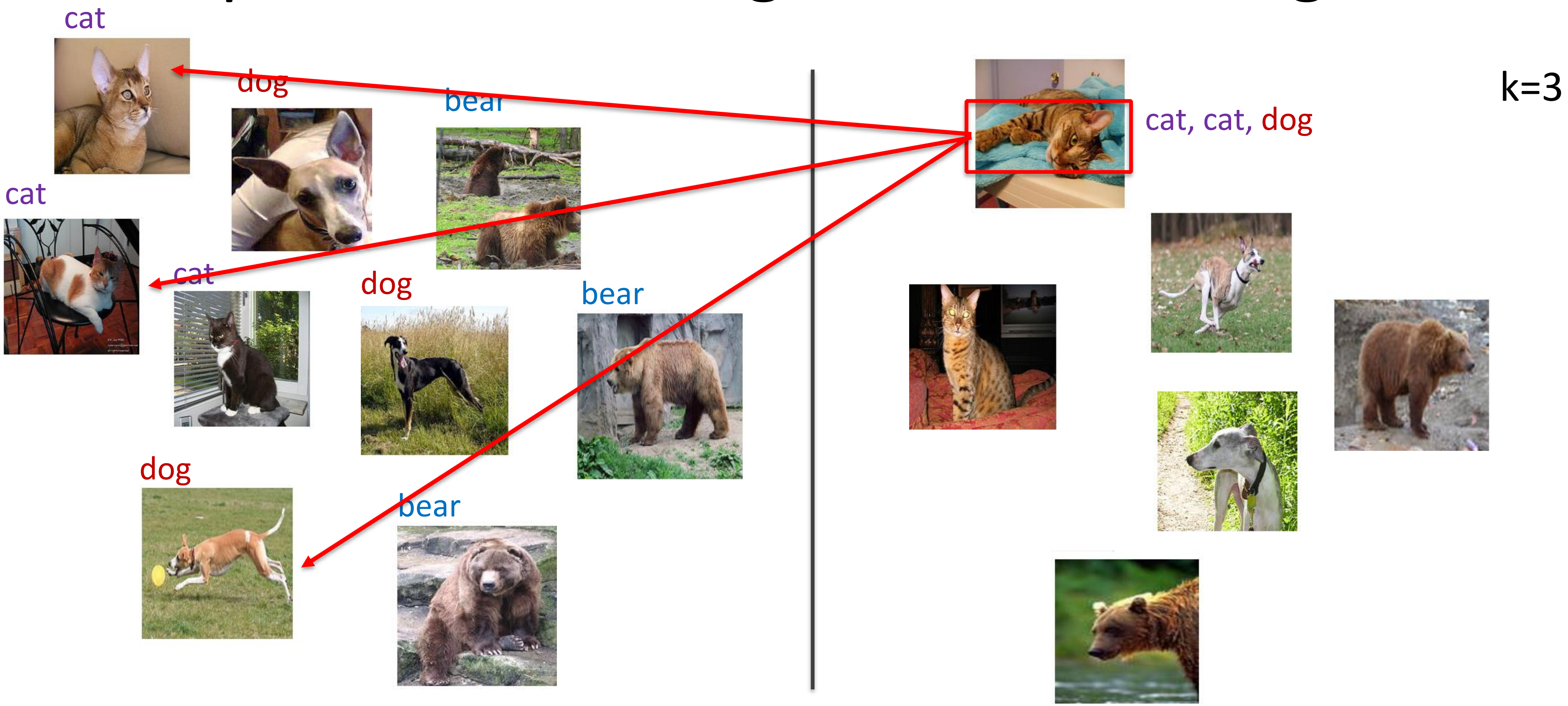
x



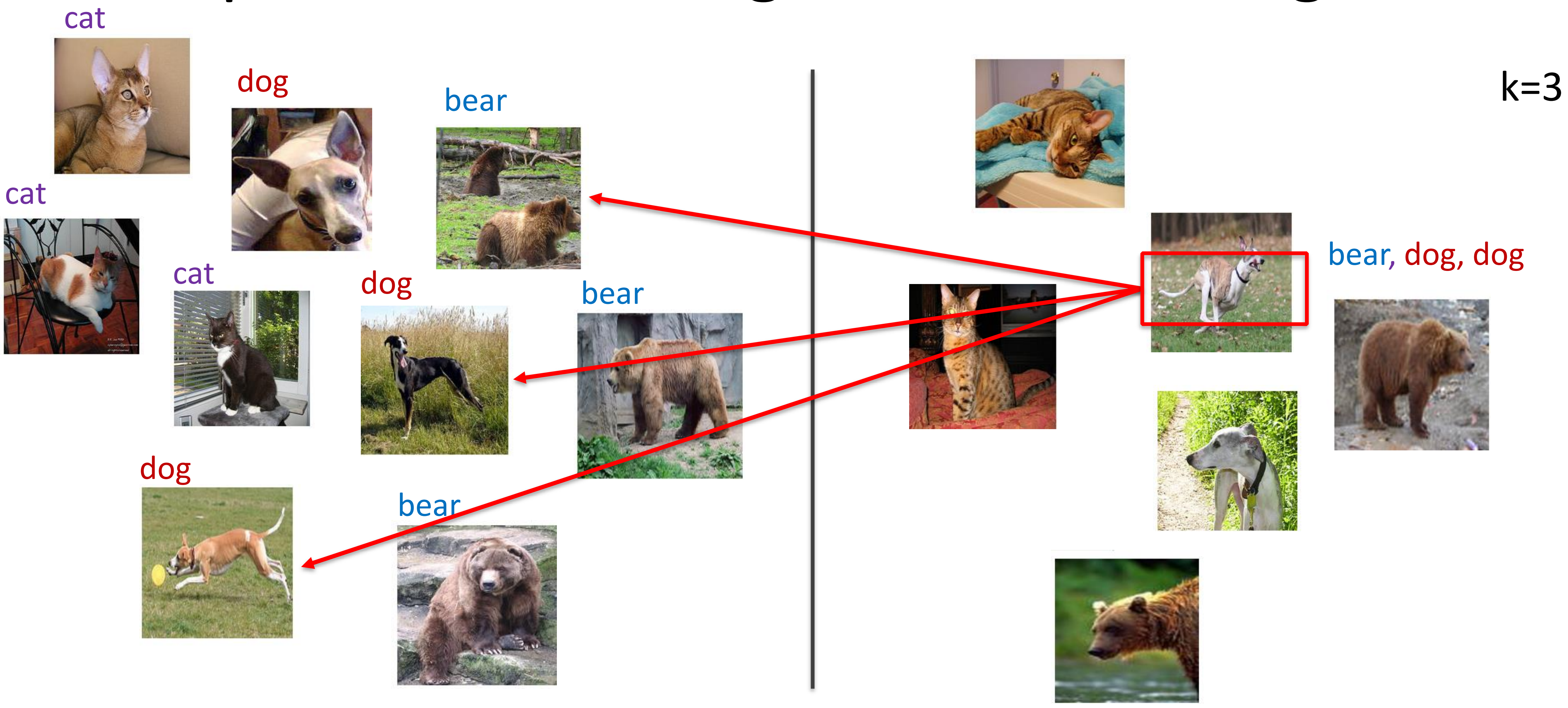
Supervised Learning vs Unsupervised Learning



Supervised Learning – k-Nearest Neighbors



Supervised Learning – k-Nearest Neighbors



Supervised Learning – k-Nearest Neighbors

- How do we choose the right K?
- How do we choose the right features?
- How do we choose the right distance metric?

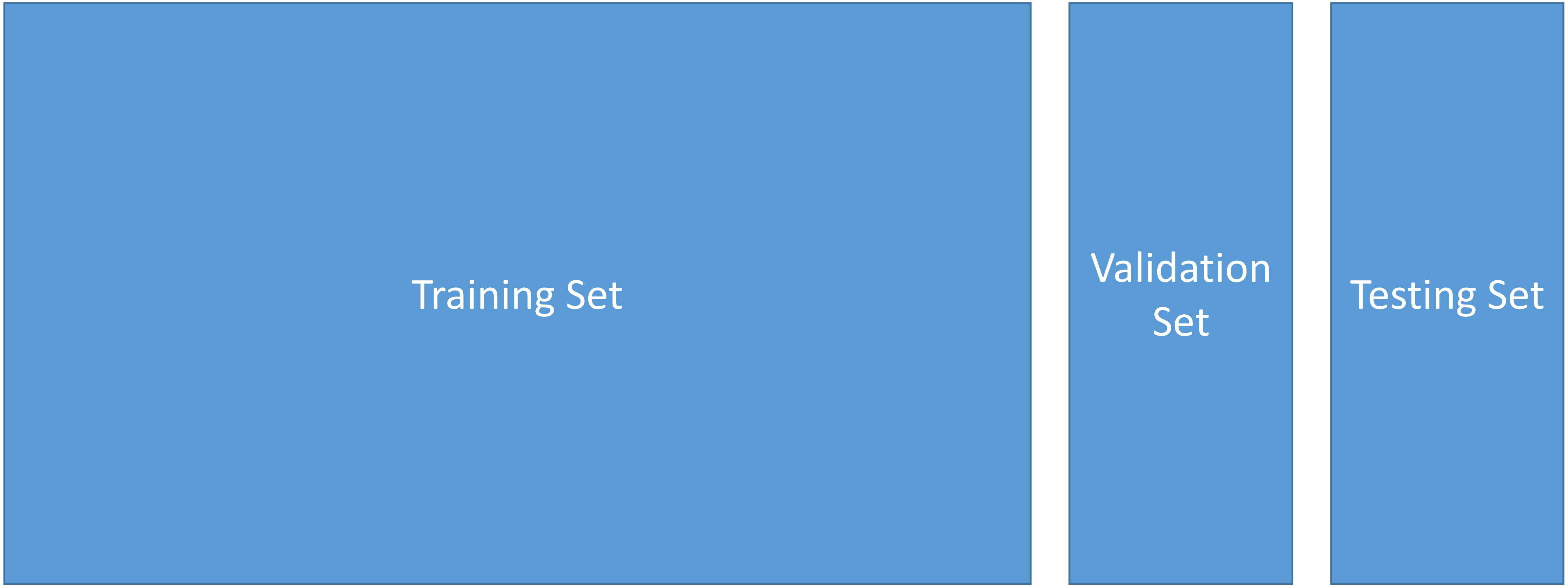
Supervised Learning – k-Nearest Neighbors

- How do we choose the right K?
- How do we choose the right features?
- How do we choose the right distance metric?

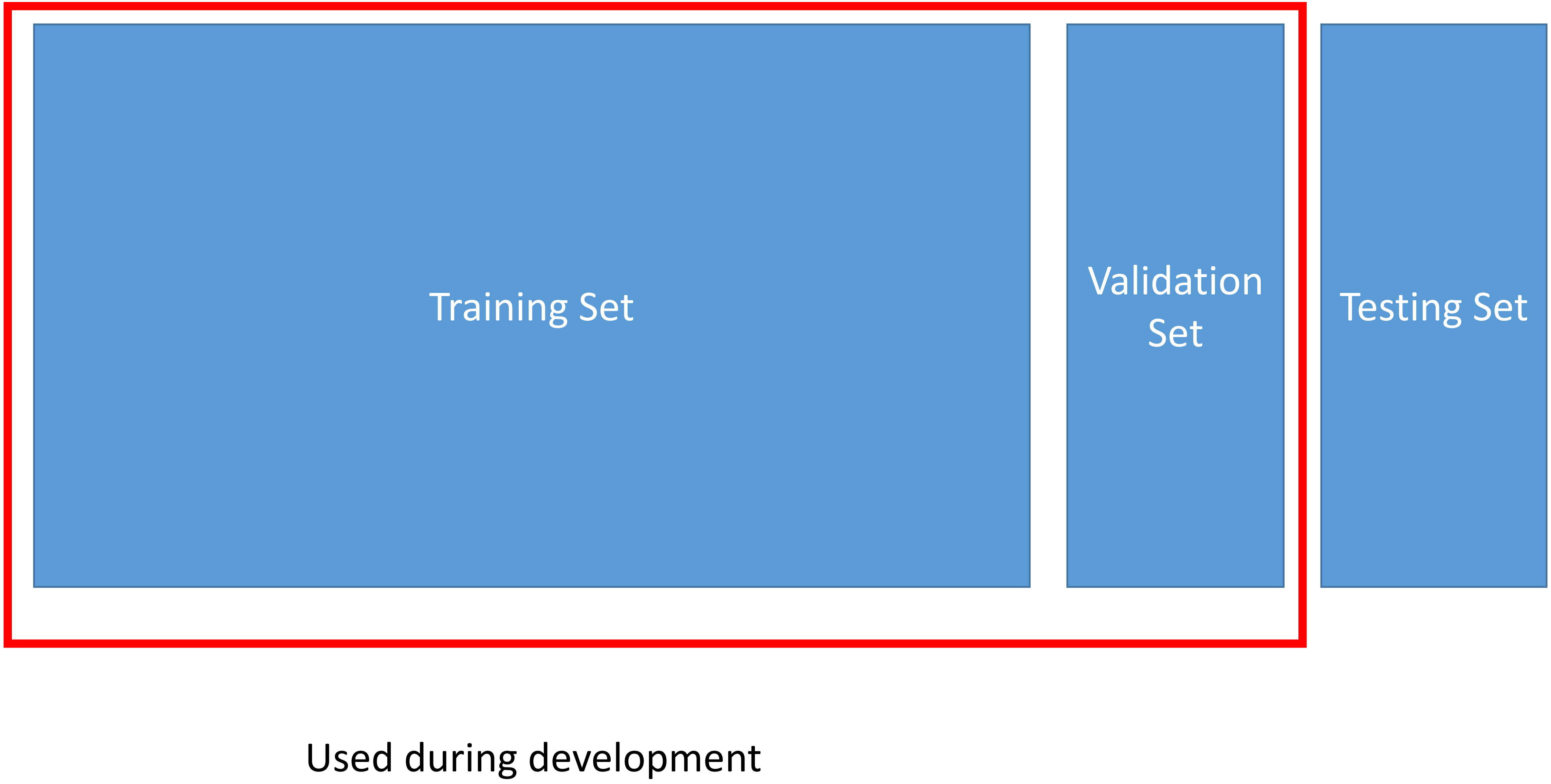
Answer: Just choose the one combination that works best!
BUT not on the test data.

Instead split the training data into a ”Training set” and a ”Validation set” (also called ”Development set”)

Training, Validation (Dev), Test Sets



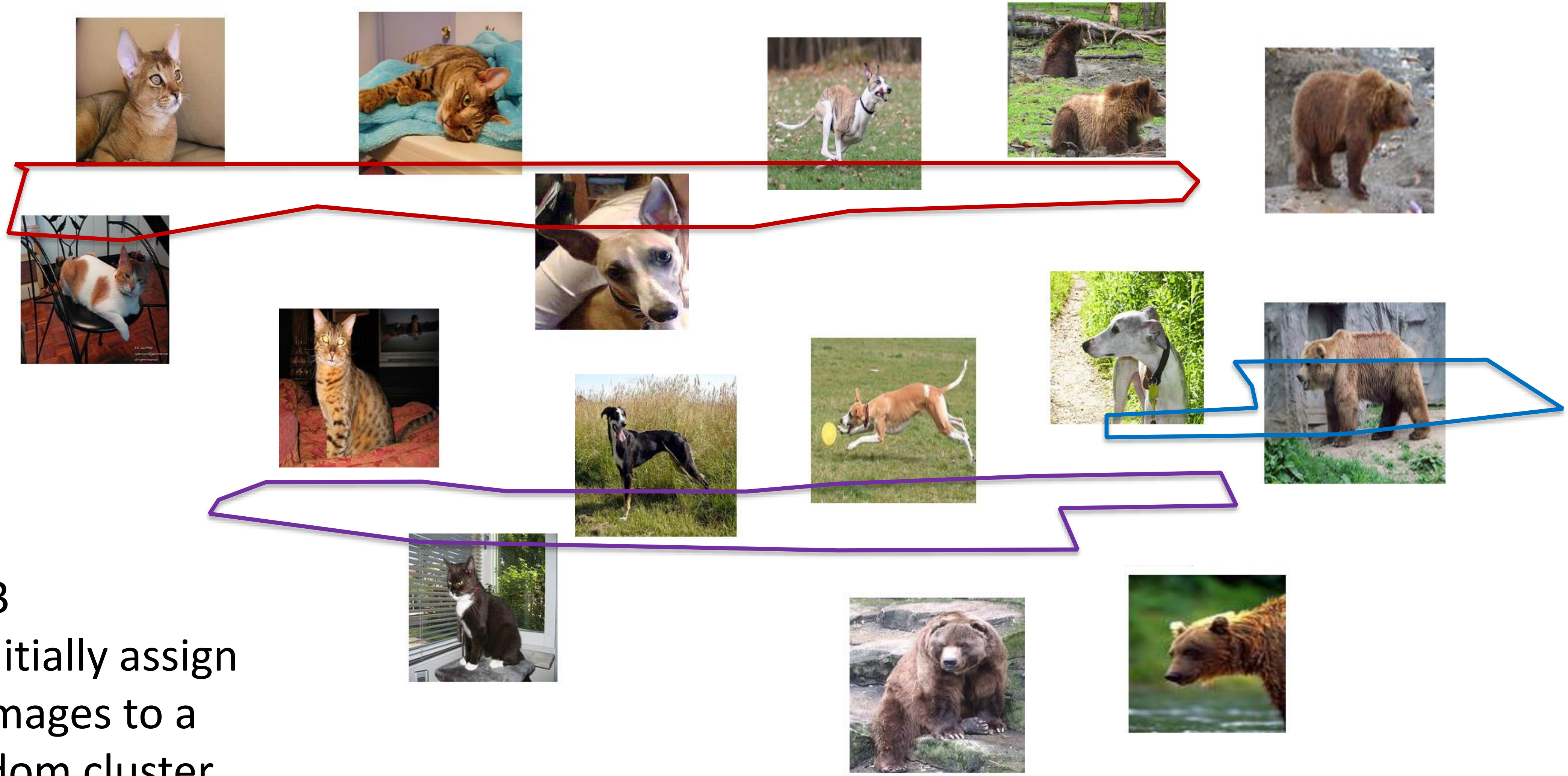
Training, Validation (Dev), Test Sets



Training, Validation (Dev), Test Sets

Only to be used for evaluating the model at the very end of development and any changes to the model after running it on the test set, could be influenced by what you saw happened on the test set, which would invalidate any future evaluation.

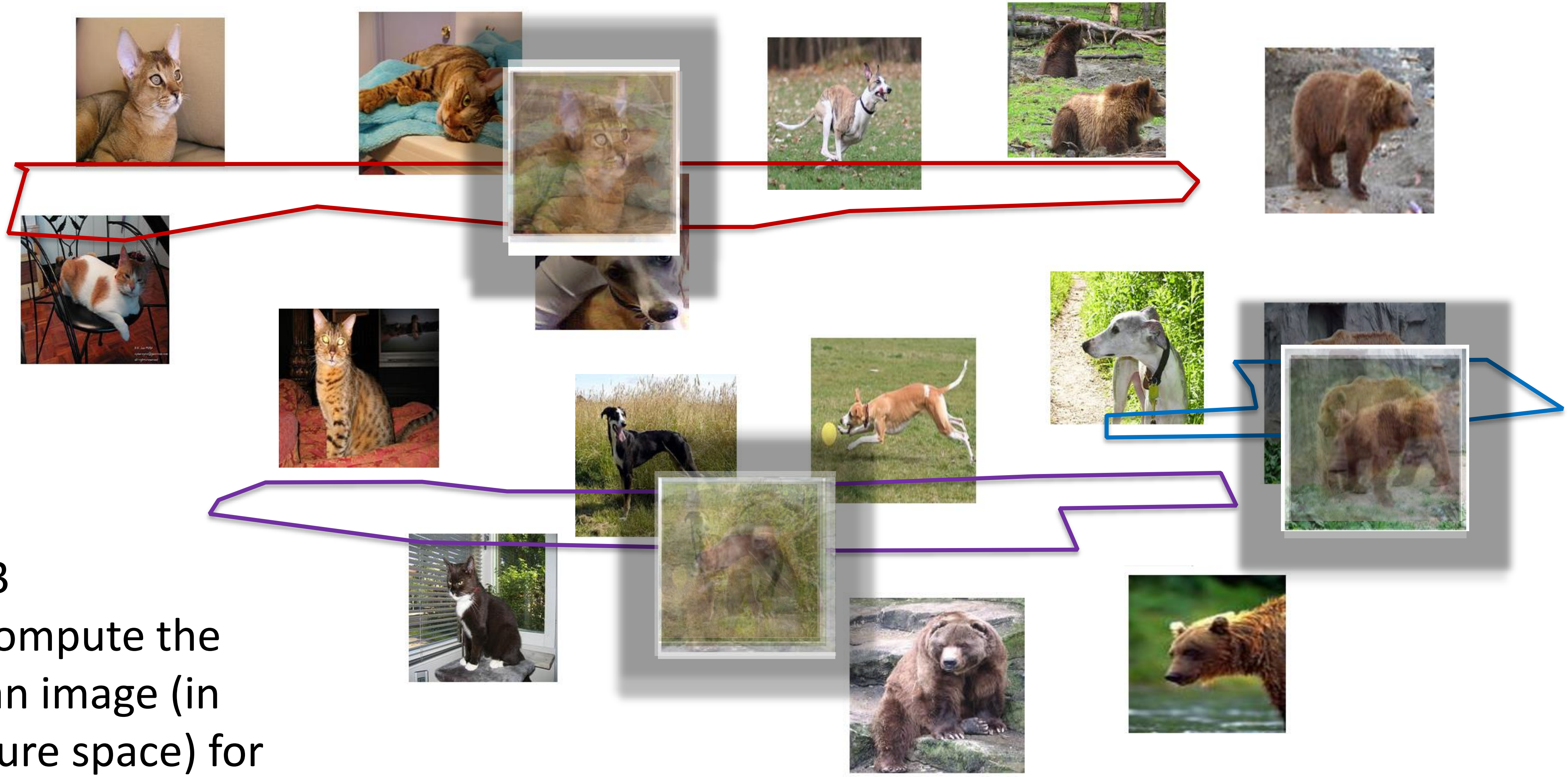
Unsupervised Learning – k-means clustering



k = 3

1. Initially assign all images to a random cluster

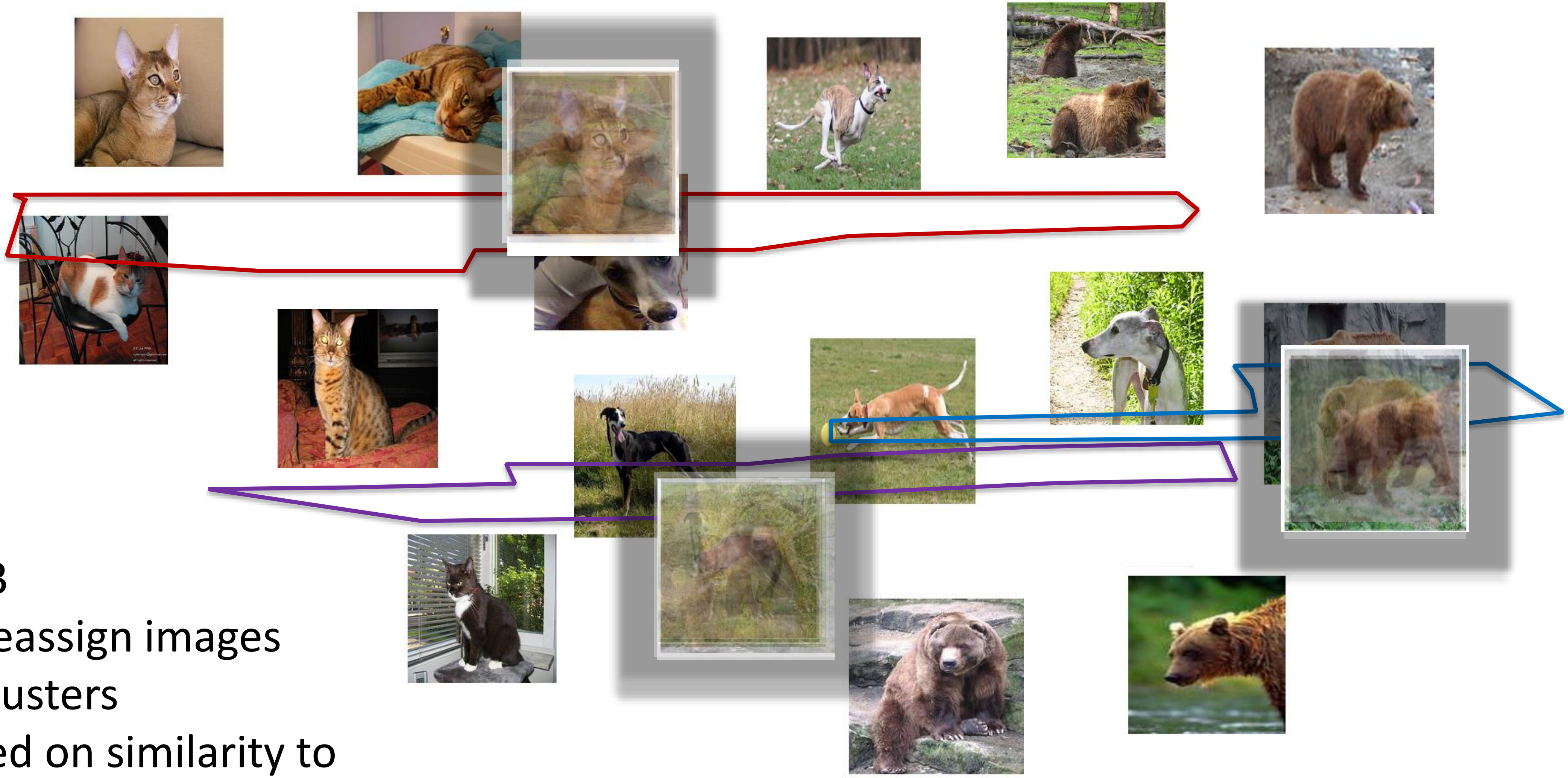
Unsupervised Learning – k-means clustering



k = 3

2. Compute the mean image (in feature space) for each cluster

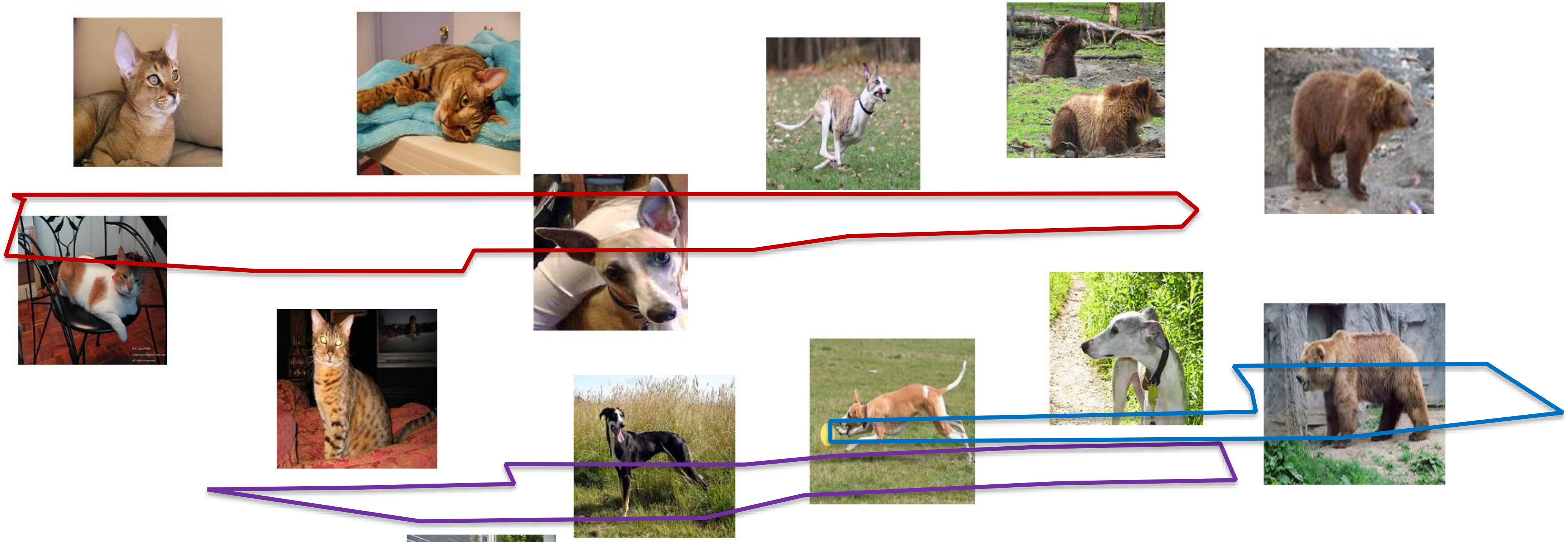
Unsupervised Learning – k-means clustering



k = 3

3. Reassign images to clusters based on similarity to cluster means

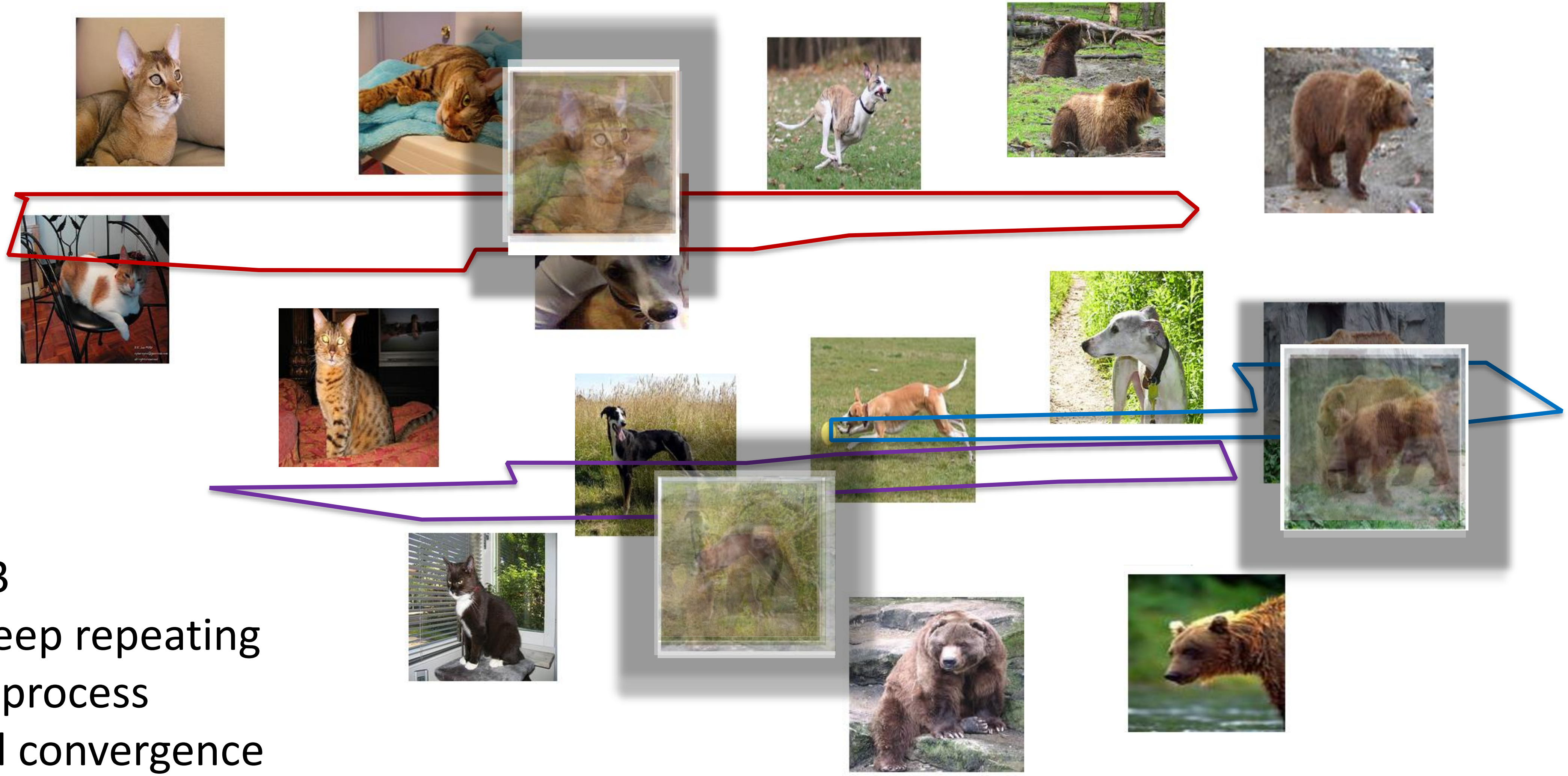
Unsupervised Learning – k-means clustering



k = 3

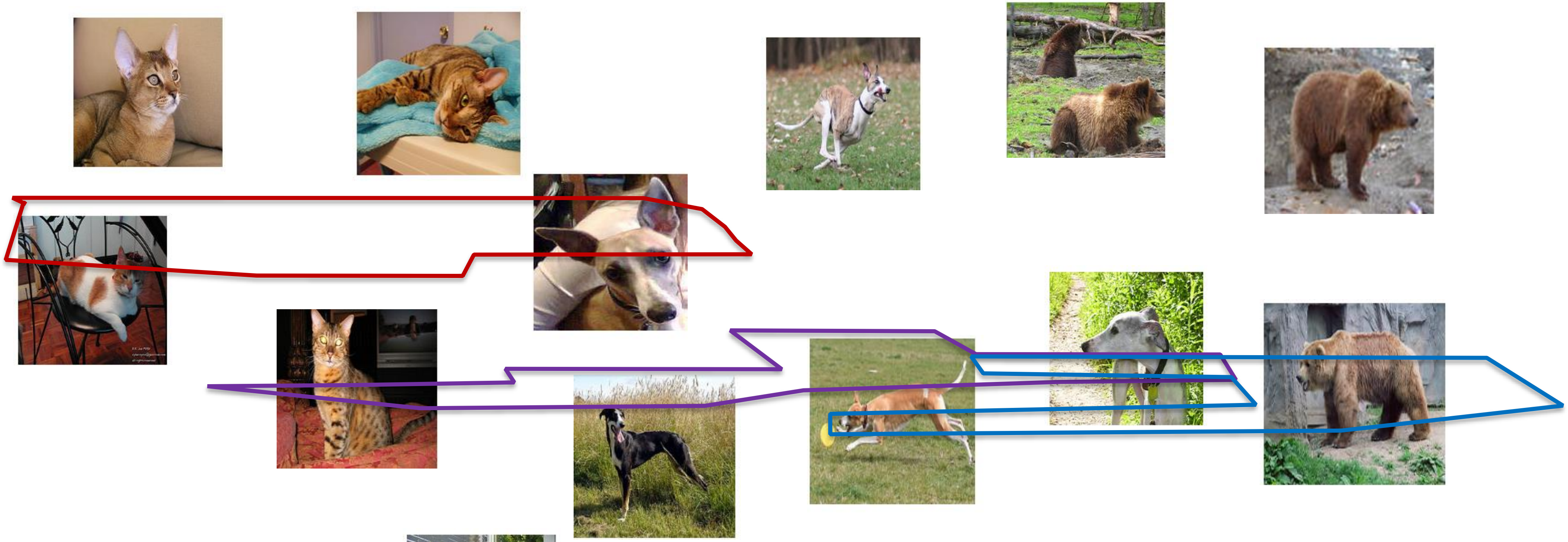
4. Keep repeating this process until convergence

Unsupervised Learning – k-means clustering



k = 3
4. Keep repeating this process until convergence

Unsupervised Learning – k-means clustering



k = 3

4. Keep repeating this process until convergence

Unsupervised Learning – k-means clustering

- How do we choose the right K?
- How do we choose the right features?
- How do we choose the right distance metric?
- How sensitive is this method with respect to the random assignment of clusters?

Supervised Learning - Classification

Training Data

cat

dog

bear

cat

cat

dog

bear

dog

bear

Test Data

Supervised Learning - Classification

Training Data

cat

dog

cat

⋮

bear

Test Data

⋮

Supervised Learning - Classification

Training Data

$x_1 = [$  $]$ $y_1 = [\text{cat}]$

$x_2 = [$  $]$ $y_2 = [\text{dog}]$

$x_3 = [$  $]$ $y_3 = [\text{cat}]$

$\vdots$

$x_n = [$  $]$ $y_n = [\text{bear}]$

Supervised Learning - Classification

Training Data

inputs	targets / labels / ground truth	predictions
$x_1 = [x_{11} \quad x_{12} \quad x_{13} \quad x_{14}]$	$y_1 = 1$	$\hat{y}_1 = 1$
$x_2 = [x_{21} \quad x_{22} \quad x_{23} \quad x_{24}]$	$y_2 = 2$	$\hat{y}_2 = 2$
$x_3 = [x_{31} \quad x_{32} \quad x_{33} \quad x_{34}]$	$y_3 = 1$	$\hat{y}_3 = 2$
$\vdots$		
$x_n = [x_{n1} \quad x_{n2} \quad x_{n3} \quad x_{n4}]$	$y_n = 3$	$\hat{y}_n = 1$

We need to find a function that maps *x* and *y* for any of them.

$$\hat{y}_i = f(x_i; \theta)$$

How do we "learn" the parameters of this function?

We choose ones that makes the following quantity small:

$$\sum_{i=1}^{n} Cost(\hat{y}_i, y_i)$$

Supervised Learning – Linear Softmax

Training Data

inputs

targets / labels / ground truth

$$x_1 = [x_{11} \quad x_{12} \quad x_{13} \quad x_{14}] \qquad y_1 = 1$$

$$x_2 = [x_{21} \quad x_{22} \quad x_{23} \quad x_{24}] \qquad y_2 = 2$$

$$x_3 = [x_{31} \quad x_{32} \quad x_{33} \quad x_{34}] \qquad y_3 = 1$$

$$\vdots$$

$$x_n = [x_{n1} \quad x_{n2} \quad x_{n3} \quad x_{n4}] \qquad y_n = 3$$

Supervised Learning – Linear Softmax

Training Data

inputs	targets / labels / ground truth	predictions
$x_1 = [x_{11} \quad x_{12} \quad x_{13} \quad x_{14}]$	$y_1 =$ [1 0 0]	$\hat{y}_1 =$ [0.85 0.10 0.05]
$x_2 = [x_{21} \quad x_{22} \quad x_{23} \quad x_{24}]$	$y_2 =$ [0 1 0]	$\hat{y}_2 =$ [0.20 0.70 0.10]
$x_3 = [x_{31} \quad x_{32} \quad x_{33} \quad x_{34}]$	$y_3 =$ [1 0 0]	$\hat{y}_3 =$ [0.40 0.45 0.15]
$\vdots$		
$x_n = [x_{n1} \quad x_{n2} \quad x_{n3} \quad x_{n4}]$	$y_n =$ [0 0 1]	$\hat{y}_n =$ [0.40 0.25 0.35]

Supervised Learning – Linear Softmax

$x_i = [x_{i1} \quad x_{i2} \quad x_{i3} \quad x_{i4}]$ $\qquad y_i = [1 \quad 0 \quad 0]$ $\qquad \hat{y}_i = [f_c \quad f_d \quad f_b]$

$$g_c = w_{c1}x_{i1} + w_{c2}x_{i2} + w_{c3}x_{i3} + w_{c4}x_{i4} + b_c$$

$$g_d = w_{d1}x_{i1} + w_{d2}x_{i2} + w_{d3}x_{i3} + w_{d4}x_{i4} + b_d$$

$$g_b = w_{b1}x_{i1} + w_{b2}x_{i2} + w_{b3}x_{i3} + w_{b4}x_{i4} + b_b$$

$$f_c = e^{g_c}/(e^{g_c}+e^{g_d} + e^{g_b})$$

$$f_d = e^{g_d}/(e^{g_c}+e^{g_d} + e^{g_b})$$

$$f_b = e^{g_b}/(e^{g_c}+e^{g_d} + e^{g_b})$$

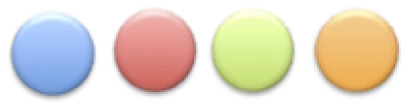
How do we find a good w and b?

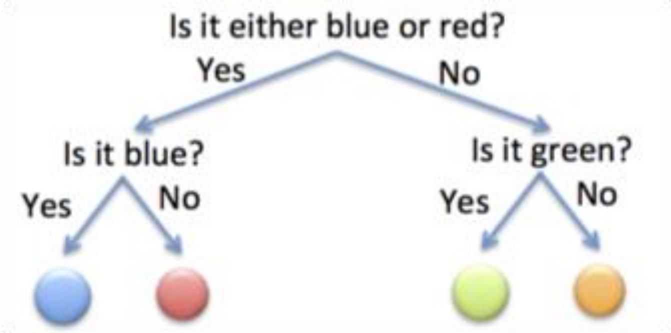
$$x_i = [x_{i1} \quad x_{i2} \quad x_{i3} \quad x_{i4}] \qquad y_i = [1 \quad 0 \quad 0] \qquad \hat{y}_i = [f_c(w,b) \quad f_d(w,b) \quad f_b(w,b)]$$

We need to find w, and b that minimize the following:

$$L(w,b) = \sum_{i=1}^{n} \sum_{j=1}^{3} -y_{i,j} \log(\hat{y}_{i,j}) \quad = \sum_{i=1}^{n} -\log(\hat{y}_{i,label}) \quad = \sum_{i=1}^{n} -\log f_{i,label}(w,b)$$

Idea of entropy and cross-entropy

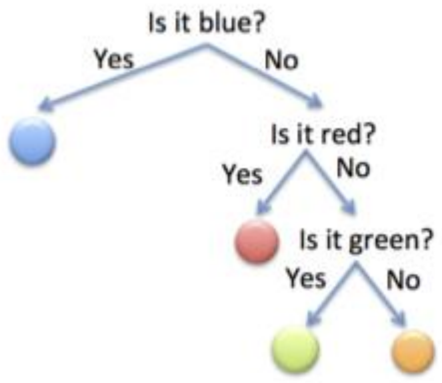




How many questions are to be
Asked to right guess the color
Of a randomly picked ball?

Another case

Now, I will draw a coin from a bag of coins: 1/2 of them are blue, 1/4 are red, 1/8 are green, and 1/8 are orange. The previous strategy no longer is the best; because there is a fair chance to draw a blue coin, we should prioritize guessing the most likely outcome. Your optimal strategy now looks like this:



Gradient Descent (GD)

$\lambda = 0.01$

Initialize w and b randomly

for e = 0, num_epochs **do**

Compute: $\underbrace{dL(w,b)/dw}$ and $\underbrace{dL(w,b)/db}$

Update w: $w = w - \lambda\, dL(w,b)/dw$

Update b: $b = b - \lambda\, dL(w,b)/db$

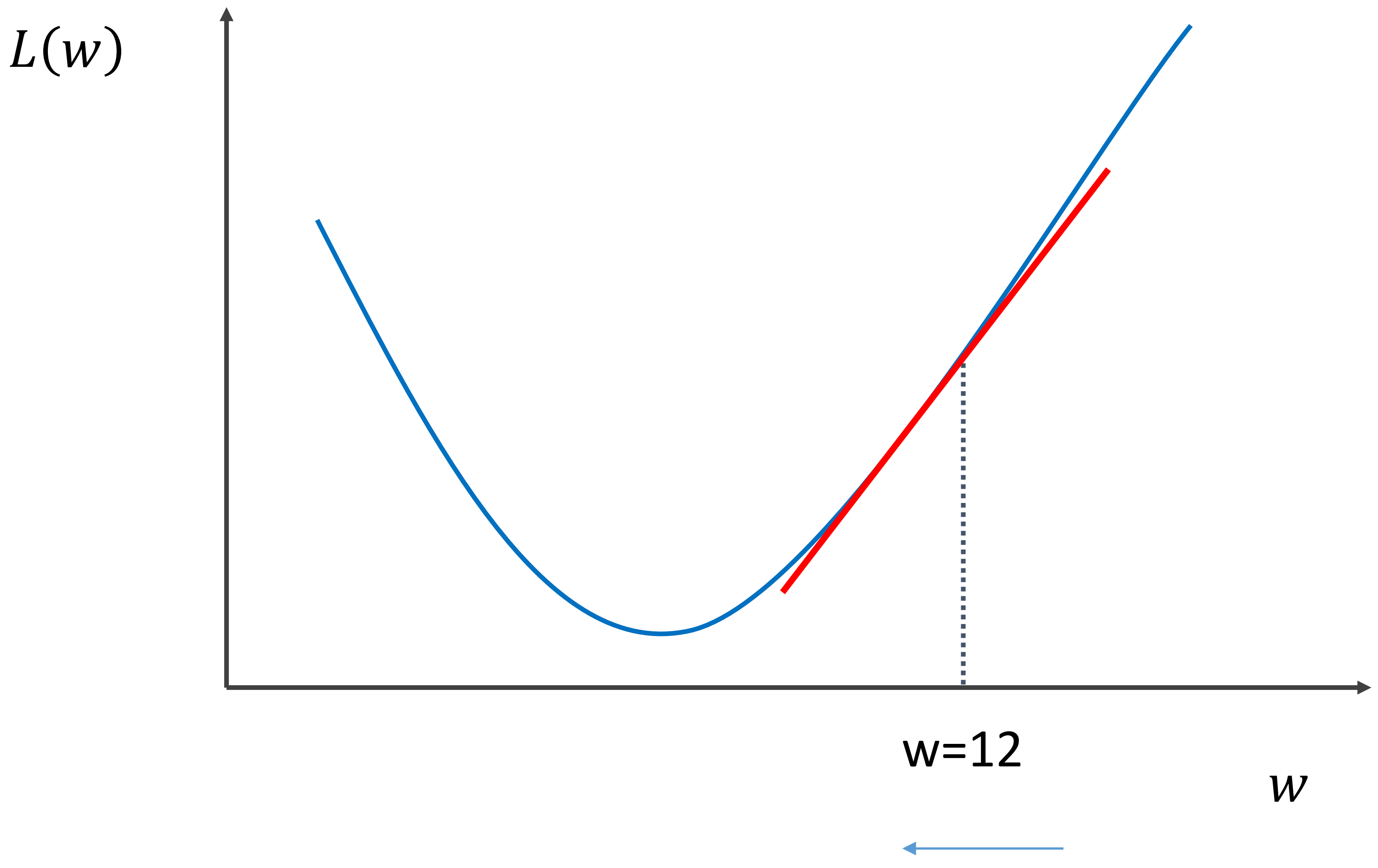
Print: $L(w,b)$ // Useful to see if this is becoming smaller or not.

end

expensive

$$L(w,b) = \sum_{i=1}^{n} -\log f_{i,label}(w,b)$$

Gradient Descent (GD) (idea)



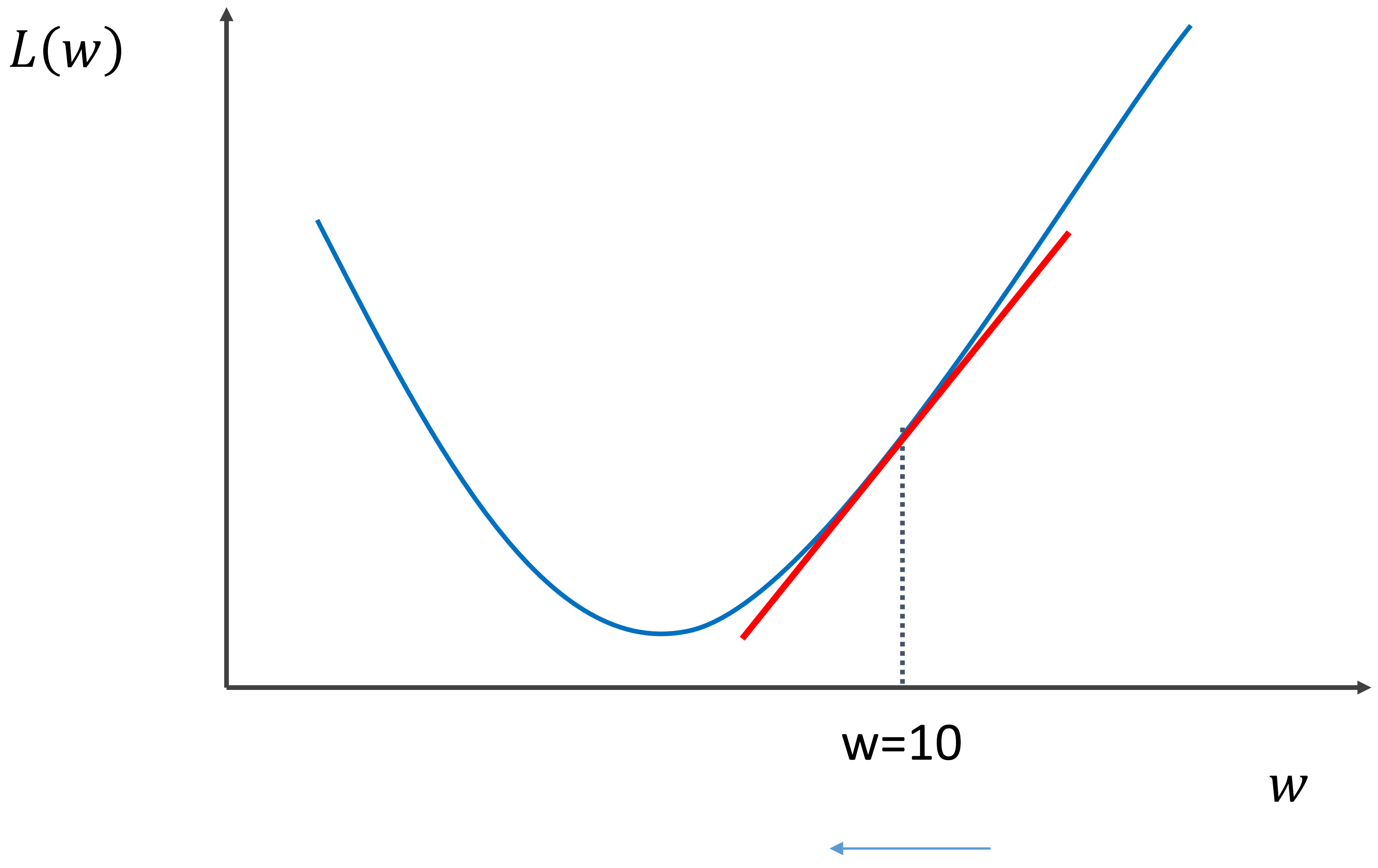
1. Start with a random value of w (e.g. w = 12)

2. Compute the gradient (derivative) of L(w) at point w = 12. (e.g. dL/dw = 6)

3. Recompute w as:

w = w – lambda * (dL / dw)

Gradient Descent (GD) (idea)



2. Compute the gradient (derivative) of L(w) at point w = 12. (e.g. dL/dw = 6)

3. Recompute w as:

w = w – lambda * (dL / dw)

(mini-batch) Stochastic Gradient Descent (SGD)

$\lambda = 0.01$

$$l(w, b) = \sum_{i \in B} -\log f_{i,label}(w, b)$$

Initialize w and b randomly

for e = 0, num_epochs **do**

for b = 0, num_batches **do**

Compute: $dl(w, b)/dw$ and $dl(w, b)/db$

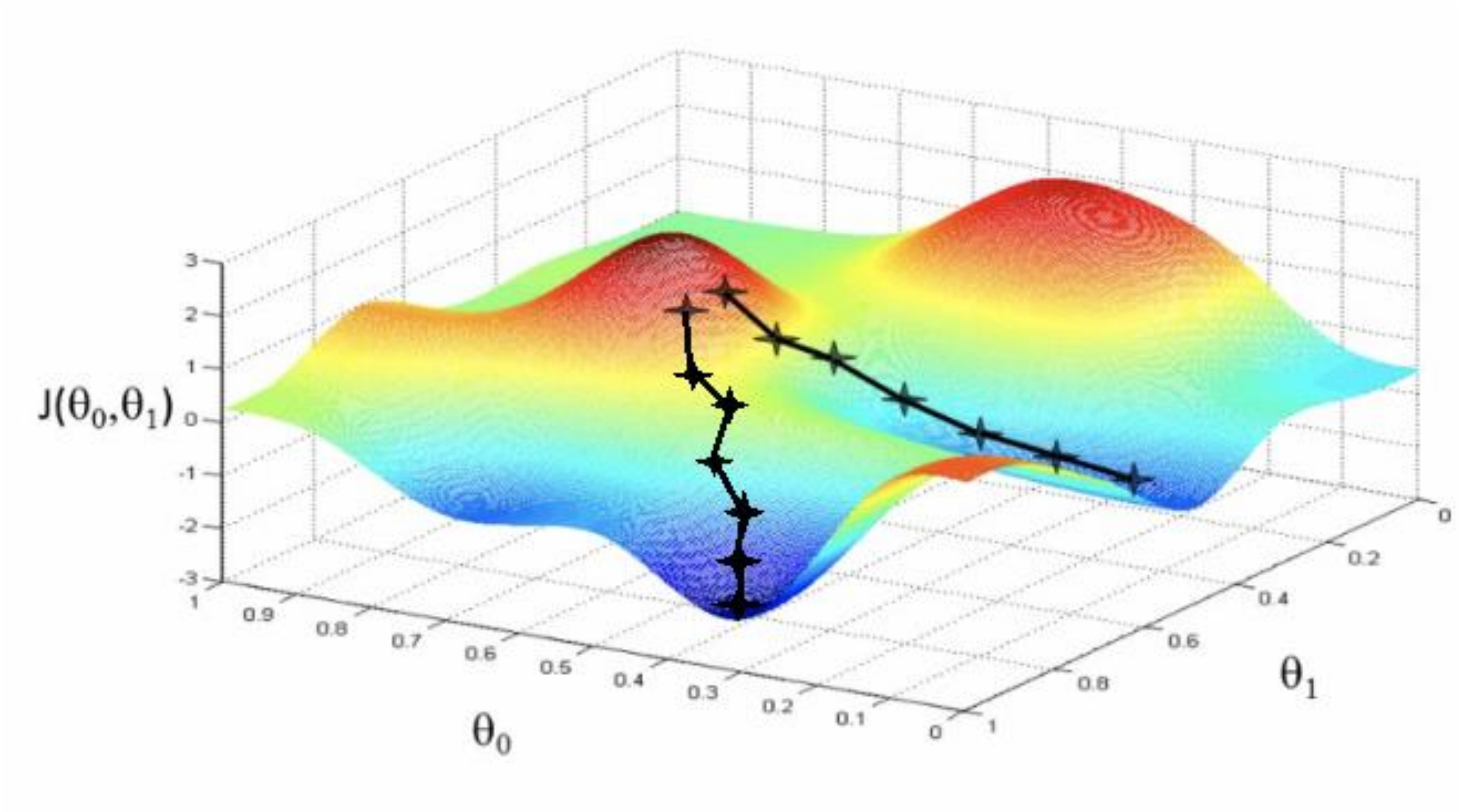
Update w: $w = w - \lambda\, dl(w, b)/dw$

Update b: $b = b - \lambda\, dl(w, b)/db$

Print: $l(w, b)$ // Useful to see if this is becoming smaller or not.

end

end



Source: Andrew Ng