Image Processing & Vision

Lecture 08: Classification II

Hak Gu Kim

hakgukim@cau.ac.kr

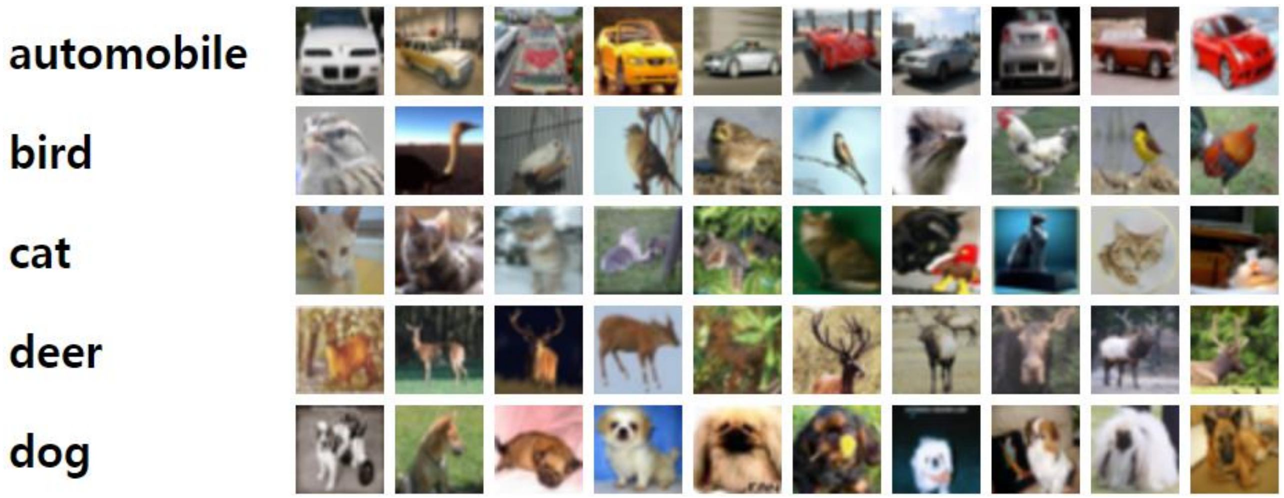
Immersive Reality & Intelligent Systems Lab (IRIS LAB)

Graduate School of Advanced Imaging Science, Multimedia & Film (GSAIM)

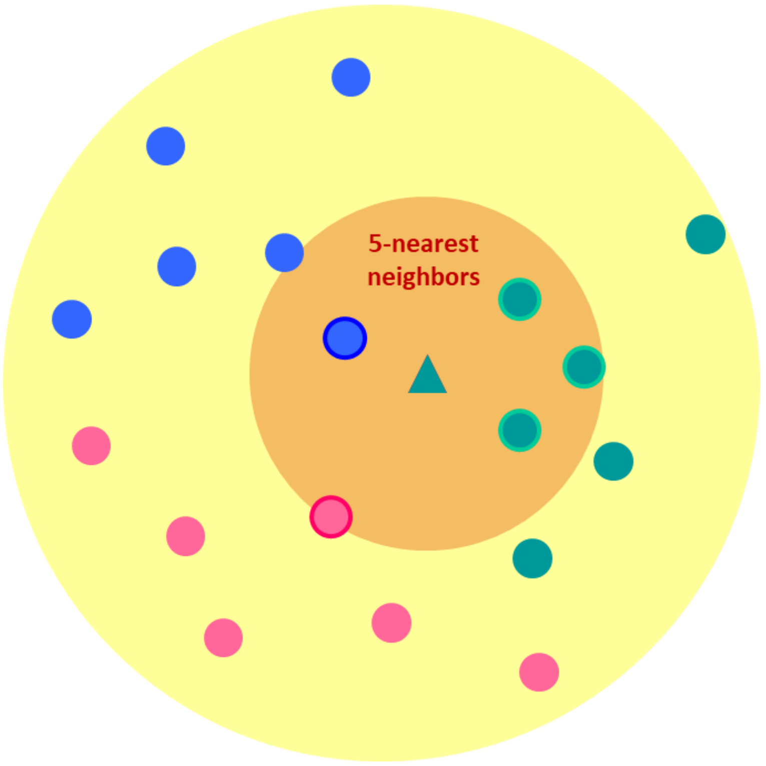
Chung-Ang University (CAU)

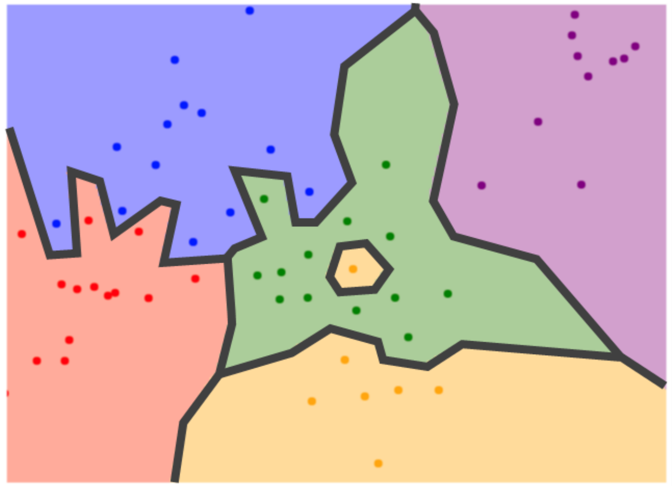
8 May 2023

Recap: Data-driven Image Classification, KNN

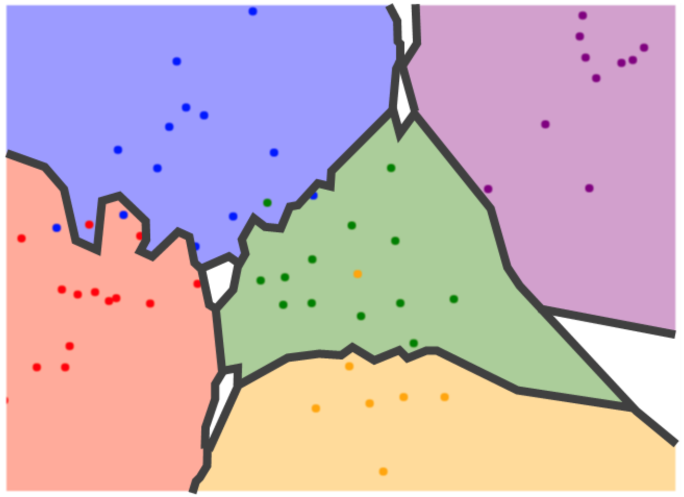


CIFAR-10 dataset for image classification

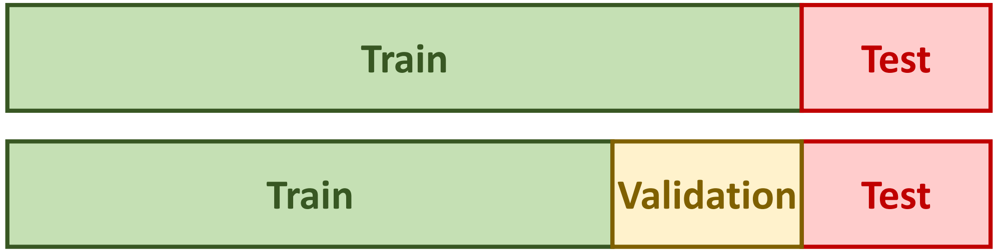




NN classifier



K-NN classifier



Split data into train/validation/test

Recap: Data-driven Image Classification, KNN

- **Disadvantages of KNN:**

— The classifier **must remember all of the training data and store it** for future comparisons with the test data. This is space inefficient because datasets may easily be gigabytes in size

— **Classifying a test image is expensive** since it requires a comparison to all training images

Topics

- Classification
— Linear Classifier with Images
— Image Features for Robust Image Classification
— Decision Tree

***Note:** Many of these slides in this course were adapted from Convolutional Neural Networks for Visual Recognition (Stanford Univ.) and Deep Learning for Computer Vision (Univ. of Michigan)

Topics

- Classification
  - Linear Classifier with Images
  - Image Features for Robust Image Classification
  - Decision Tree

***Note:** Many of these slides in this course were adapted from Convolutional Neural Networks for Visual Recognition (Stanford Univ.) and Deep Learning for Computer Vision (Univ. of Michigan)

Parameterized Mapping: From Images to Label Scores

- The score function, f, is defined to map the pixel values of an image to confidence scores for each class

Image, $\mathbf{x} \in \mathbb{R}^{32\times32\times3}$

$f(\mathbf{x}, \mathbf{W})$

Scores for each class (**10** classes)

Parameters of weights, $\mathbf{W} \in \mathbb{R}^{10\times3072}$

Parameterized Mapping: From Images to Label Scores

- The score function, f, is defined to map the pixel values of an image to confidence scores for each class
  - $\mathbf{W}$: Weights
  - $\mathbf{b}$: Bias vector

$$\underset{10\times 1}{\boxed{f(\mathbf{x},\mathbf{W})}} = \underset{10\times 3072}{\boxed{\mathbf{W}}}\overset{3072\times 1}{\boxed{\mathbf{x}}} + \overset{10\times 1}{\boxed{\mathbf{b}}}$$

Image, $\mathbf{x} \in \mathbb{R}^{32\times 32\times 3}$

$$\mathbf{x} \longrightarrow f(\mathbf{x},\mathbf{W}) \longrightarrow$$

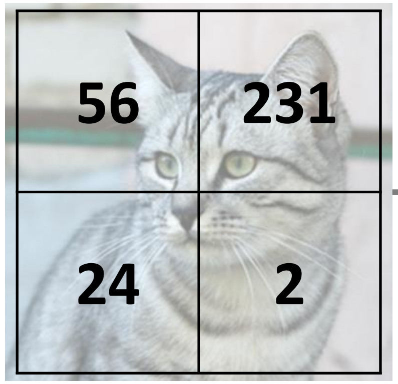
Scores for each class (**10** classes)

Parameters of weights, $\mathbf{W} \in \mathbb{R}^{10\times 3072}$

Example: 2x2 Image Classification (3 Classes)

- Classification: Cat / Dog / Bird

$$f(\mathbf{x}, \mathbf{W}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$



Input image,
$\mathbf{x} \in \mathbb{R}^{2\times2}$

Stretch pixels into column vector
(i.e., reshape)

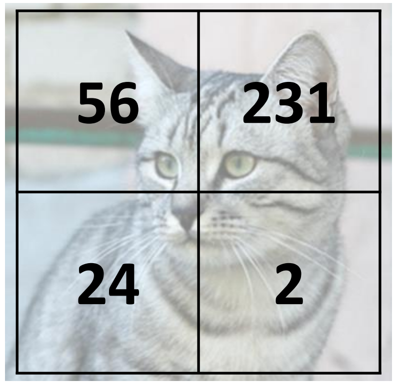
$$\begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix}$$

$\mathbf{x} \in \mathbb{R}^{4\times1}$

Example: 2x2 Image Classification (3 Classes)

- Classification: Cat / Dog / Bird

$$f(\mathbf{x}, \mathbf{W}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$



Input image,
$\mathbf{x} \in \mathbb{R}^{2\times 2}$

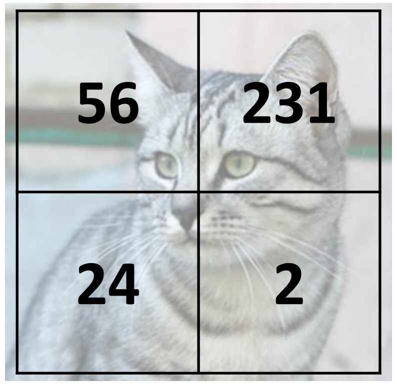
$$\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0.0 & 0.25 & 0.2 & -0.3 \end{bmatrix} \times \begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix} + \begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix} = \begin{array}{c} \begin{bmatrix} -96.8 \\ \boxed{437.9} \\ 61.95 \end{bmatrix} \end{array} \begin{array}{l} \text{Cat} \\ \text{Dog} \\ \text{Bird} \end{array}$$

$\mathbf{W} \in \mathbb{R}^{3\times 4}$ $\quad$ $\mathbf{x} \in \mathbb{R}^{4\times 1}$ $\quad$ $\mathbf{b} \in \mathbb{R}^{3\times 1}$ $\quad$ Score, $f(\mathbf{x}; \mathbf{W}, \mathbf{b})$

Example: 2x2 Image Classification (3 Classes)

- Classification: Cat / Dog / Bird

$$f(\mathbf{x}, \mathbf{W}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$



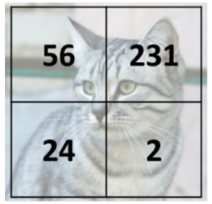
Input image, $\mathbf{x} \in \mathbb{R}^{2\times2}$

$$\begin{bmatrix} 1.2 & 1.5 & 1.1 & 2.0 \\ -0.5 & 0.3 & 0.1 & 0.0 \\ 0.1 & 0.05 & -0.1 & -0.2 \end{bmatrix} \times \begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix} + \begin{bmatrix} 3.1 \\ 1.2 \\ -1.5 \end{bmatrix} = \begin{bmatrix} \boxed{447.2} \\ 44.9 \\ 12.85 \end{bmatrix} \begin{array}{l} \text{Cat} \\ \text{Dog} \\ \text{Bird} \end{array}$$

$\mathbf{W} \in \mathbb{R}^{3\times4}$ $\quad$ $\mathbf{x} \in \mathbb{R}^{4\times1}$ $\quad$ $\mathbf{b} \in \mathbb{R}^{3\times1}$ $\quad$ Score, $f(\mathbf{x}; \mathbf{W}, \mathbf{b})$

Interpretation of Linear Classifiers: **Algebraic Viewpoint**

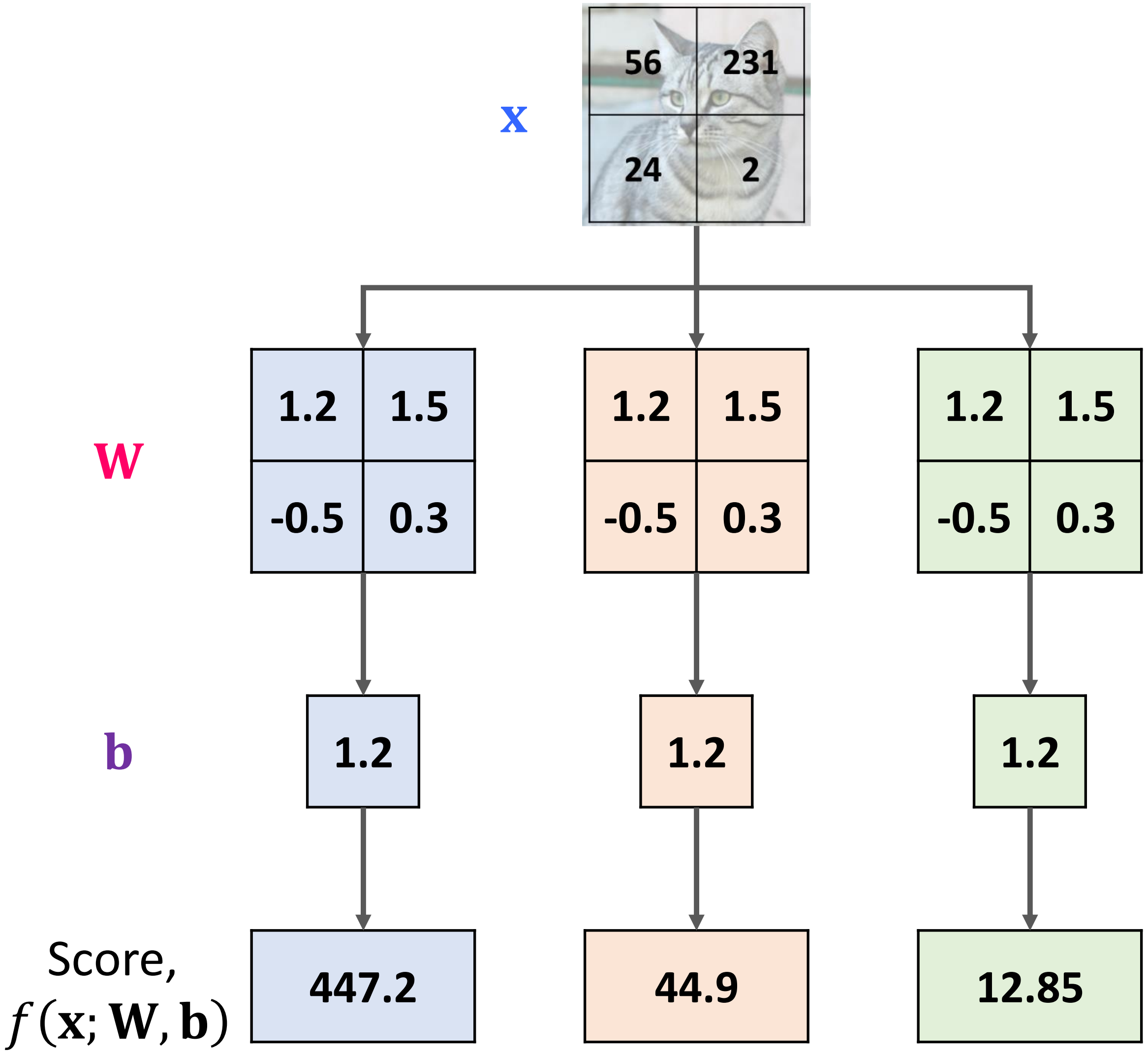
$$f(\mathbf{x}, \mathbf{W}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$



Input image, $\mathbf{x} \in \mathbb{R}^{2\times2}$

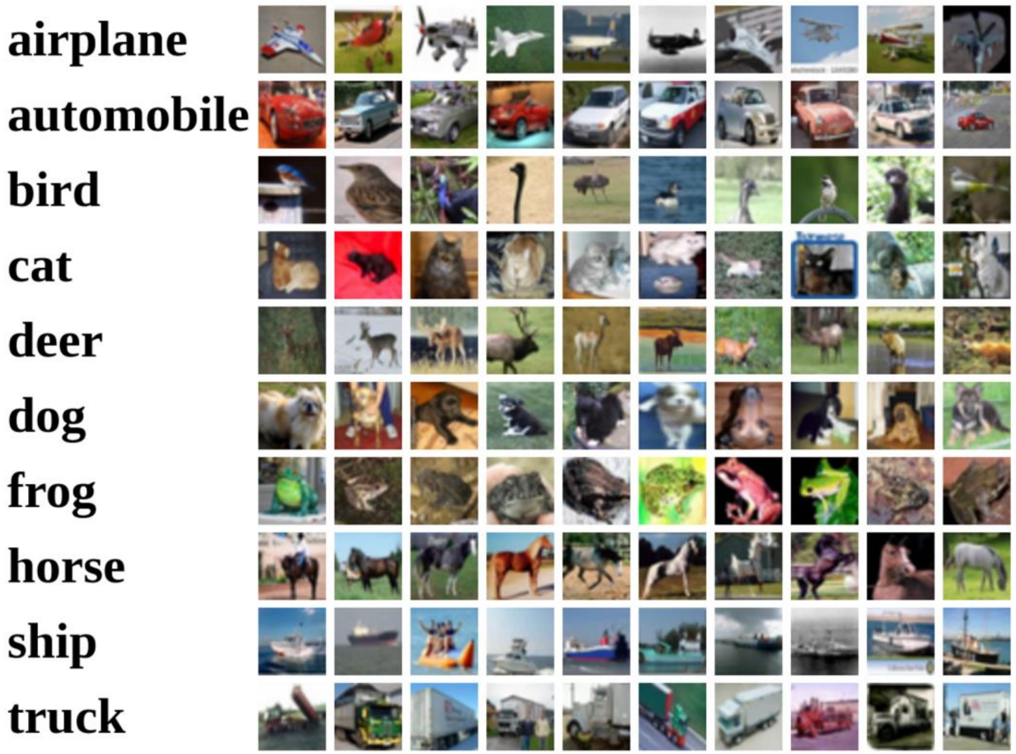
$$\begin{bmatrix} 1.2 & 1.5 & 1.1 & 2.0 \\ -0.5 & 0.3 & 0.1 & 0.0 \\ 0.1 & 0.05 & -0.1 & -0.2 \end{bmatrix} \times \begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix} + \begin{bmatrix} 3.1 \\ 1.2 \\ -1.5 \end{bmatrix} = \begin{array}{c} \begin{bmatrix} 447.2 \\ 44.9 \\ 12.85 \end{bmatrix} \end{array} \begin{array}{l} \text{Cat} \\ \text{Dog} \\ \text{Bird} \end{array}$$

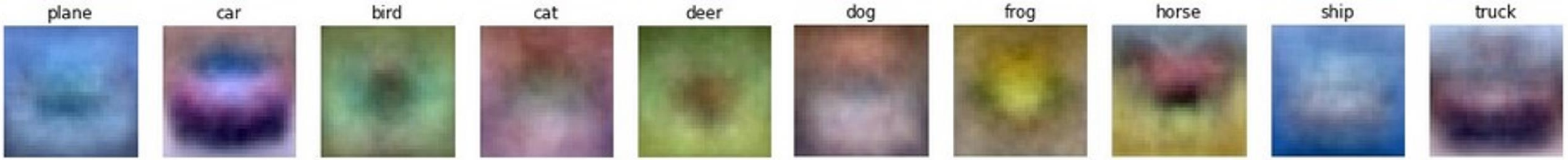
$\mathbf{W} \in \mathbb{R}^{3\times4}$ $\quad$ $\mathbf{x} \in \mathbb{R}^{4\times1}$ $\quad$ $\mathbf{b} \in \mathbb{R}^{3\times1}$ $\quad$ Score, $f(\mathbf{x}; \mathbf{W}, \mathbf{b})$



Interpretation of Linear Classifiers: **Visual Viewpoint**

- Another interpretation for the weights, **W** is that **each row of W** corresponds to **a template** for one of the classes
- The score of each class for an image is obtained **by comparing each template with the image using a dot product** one by one to find the one that "fits" best





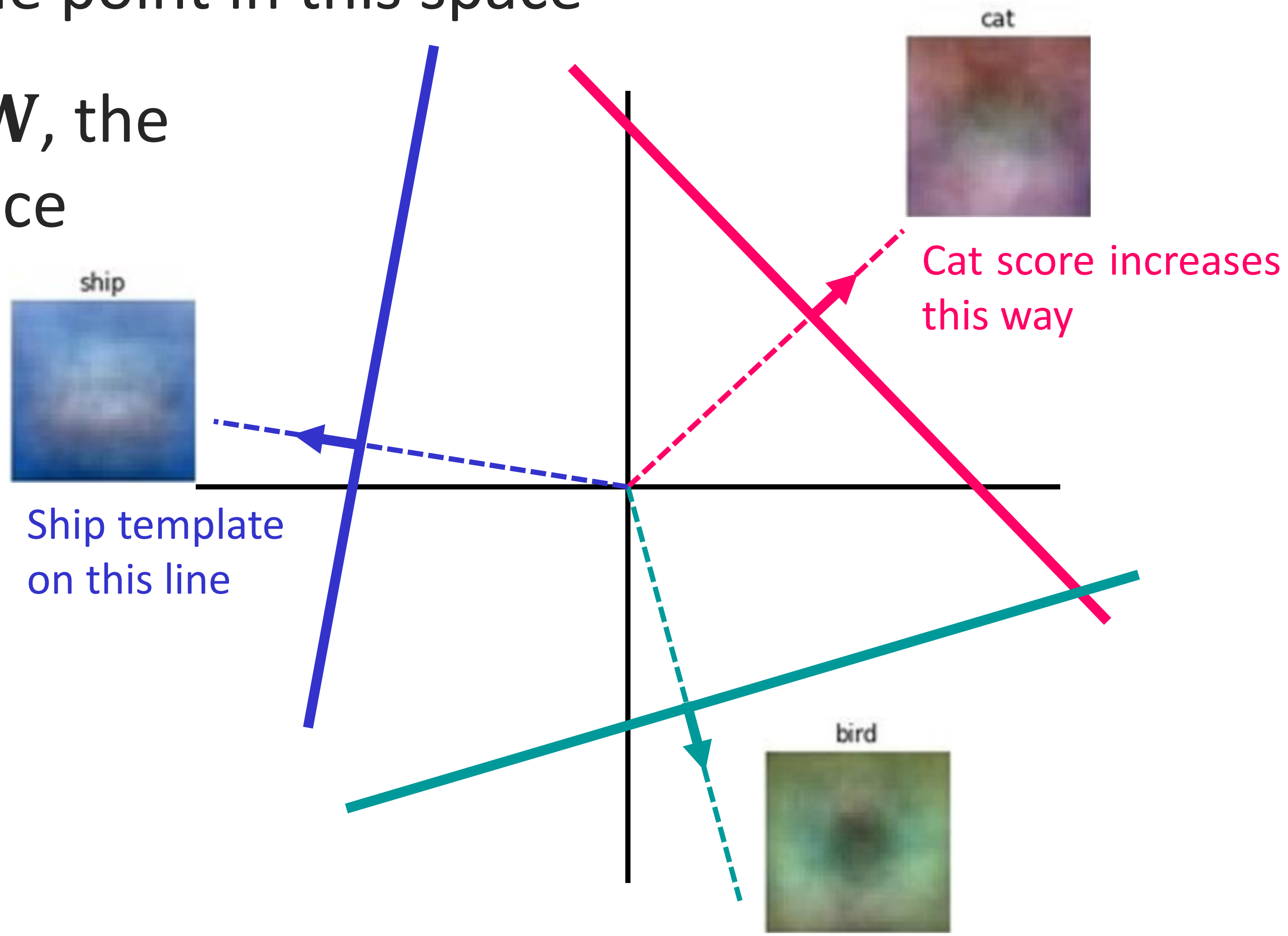
Example learned weights at the end of learning for CIFAR-10

Interpretation of Linear Classifiers: **Geometric Viewpoint**

- Since the images are stretched into high-dimensional column vectors, we can interpret each image as a single point in this space
- As we change one of the rows of $\mathbf{W}$, the corresponding line in the pixel space will rotate in different directions

$$\begin{bmatrix} 1.2 & 1.5 & 1.1 & 2.0 \\ -0.5 & 0.3 & 0.1 & 0.0 \\ 0.1 & 0.05 & -0.1 & -0.2 \end{bmatrix} \times \begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix} + \begin{bmatrix} 3.1 \\ 1.2 \\ -1.5 \end{bmatrix} = \begin{array}{c} \begin{bmatrix} 447.2 \\ 44.9 \\ 12.85 \end{bmatrix} \end{array} \begin{array}{l} \text{Cat} \\ \text{Dog} \\ \text{Bird} \end{array}$$

$\mathbf{W} \in \mathbb{R}^{3\times 4}$ $\quad$ $\mathbf{x} \in \mathbb{R}^{4\times 1}$ $\quad$ $\mathbf{b} \in \mathbb{R}^{3\times 1}$ $\quad$ Score, $f(\mathbf{x}; \mathbf{W}, \mathbf{b})$

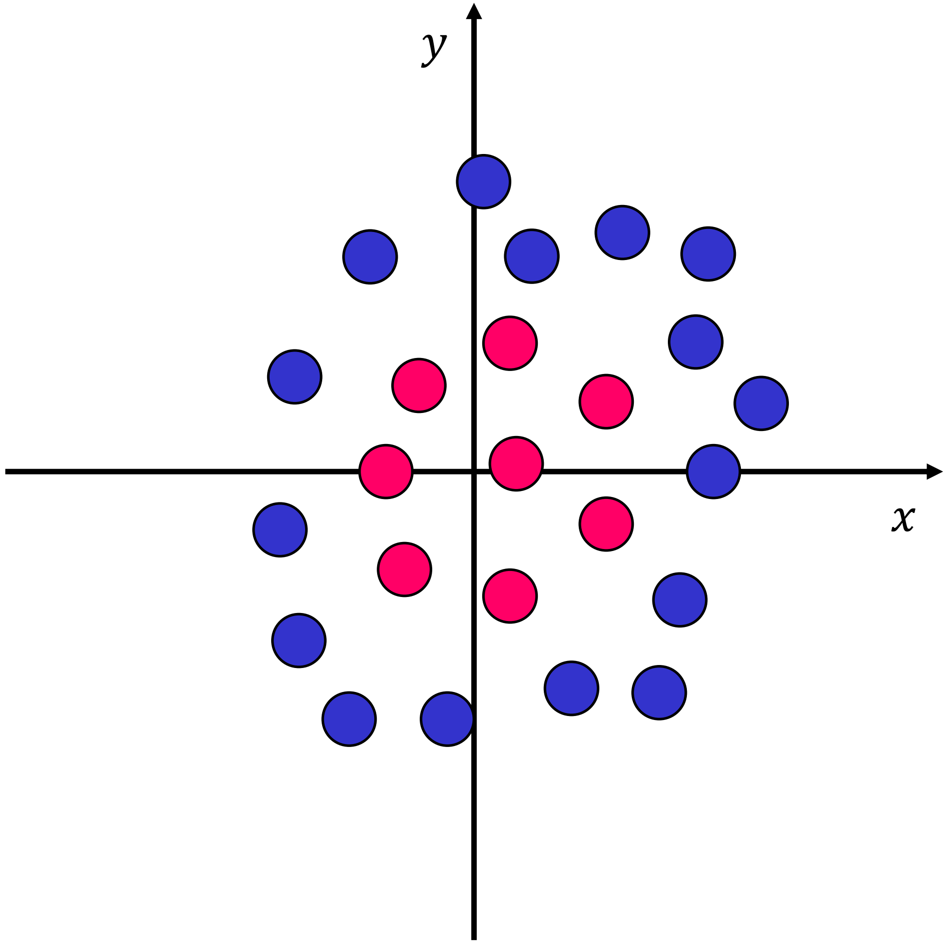


Topics

- Classification
  - Linear Classifier with Images
  - Image Features for Robust Image Classification
  - Decision Tree

Problem: **Limitation of Linear Classifiers**

- **Geometric Viewpoint**

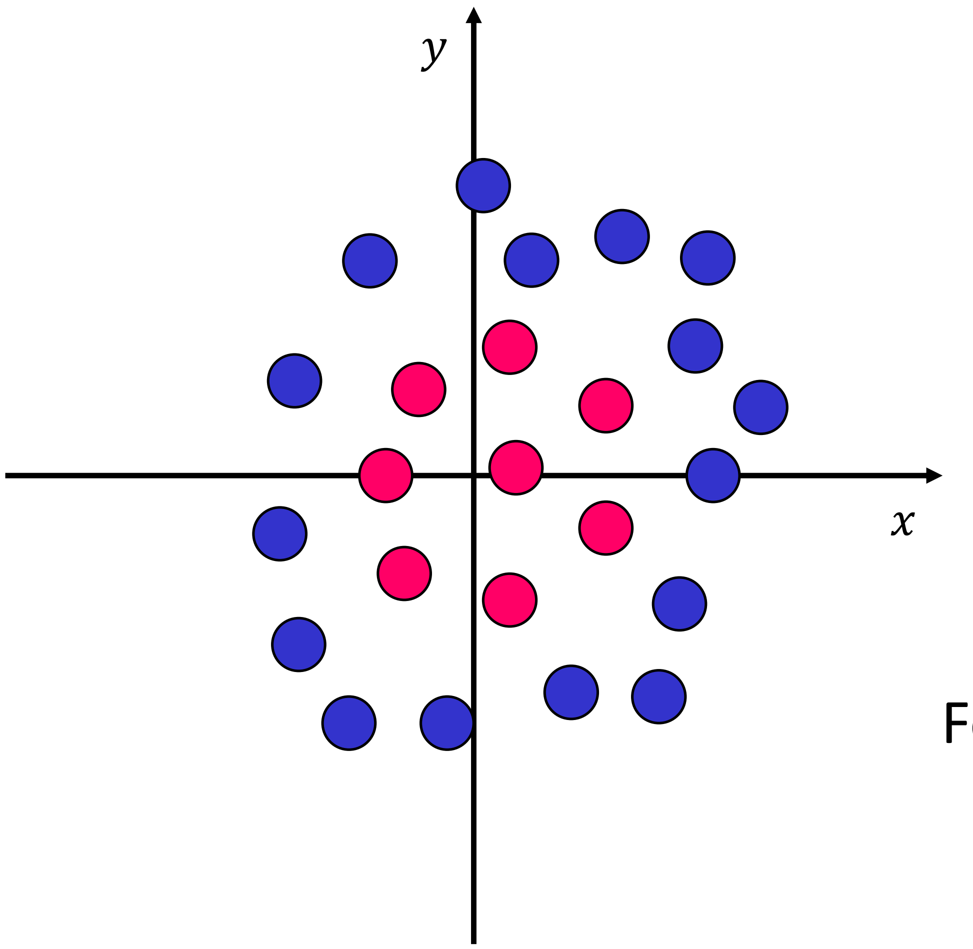


- **Visual Viewpoint**

— One template per class: it cannot recognize different modes of a class

Solution: **Feature Transforms**

Original space

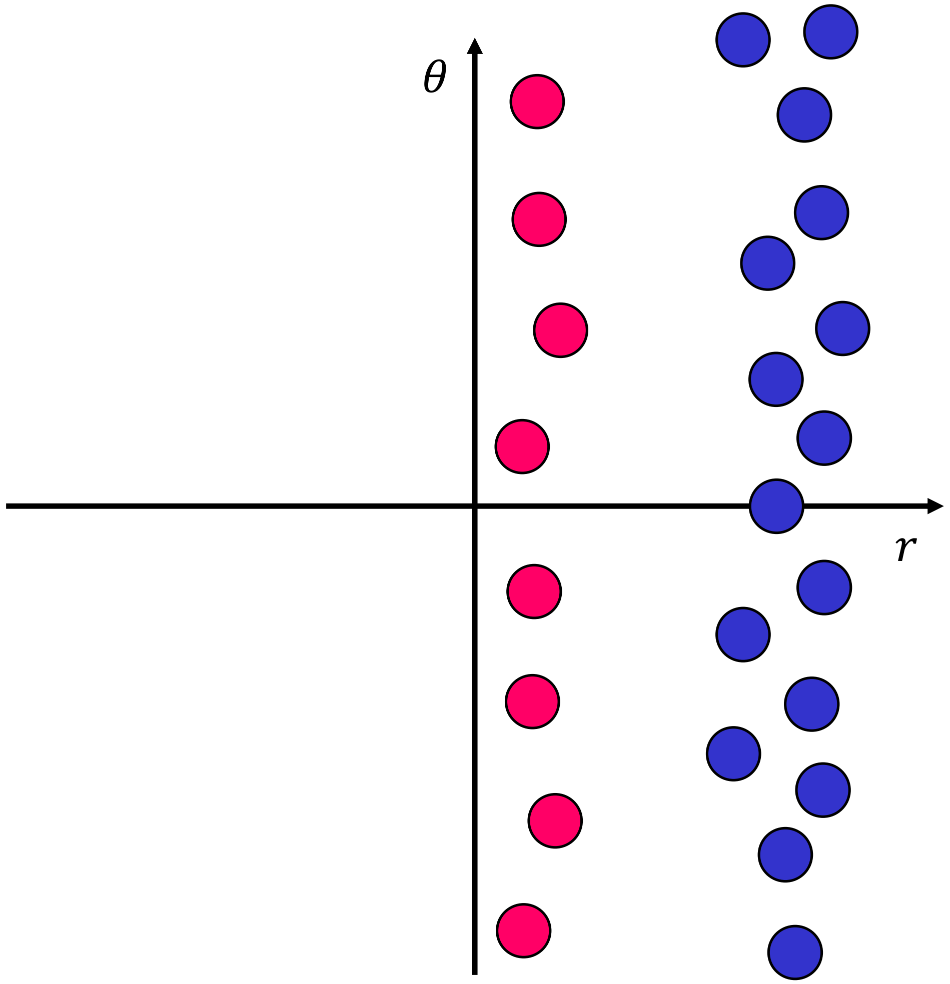


$$r = \sqrt{x^2 + y^2}$$

$$\theta = \tan^{-1}\left(\frac{y}{x}\right)$$

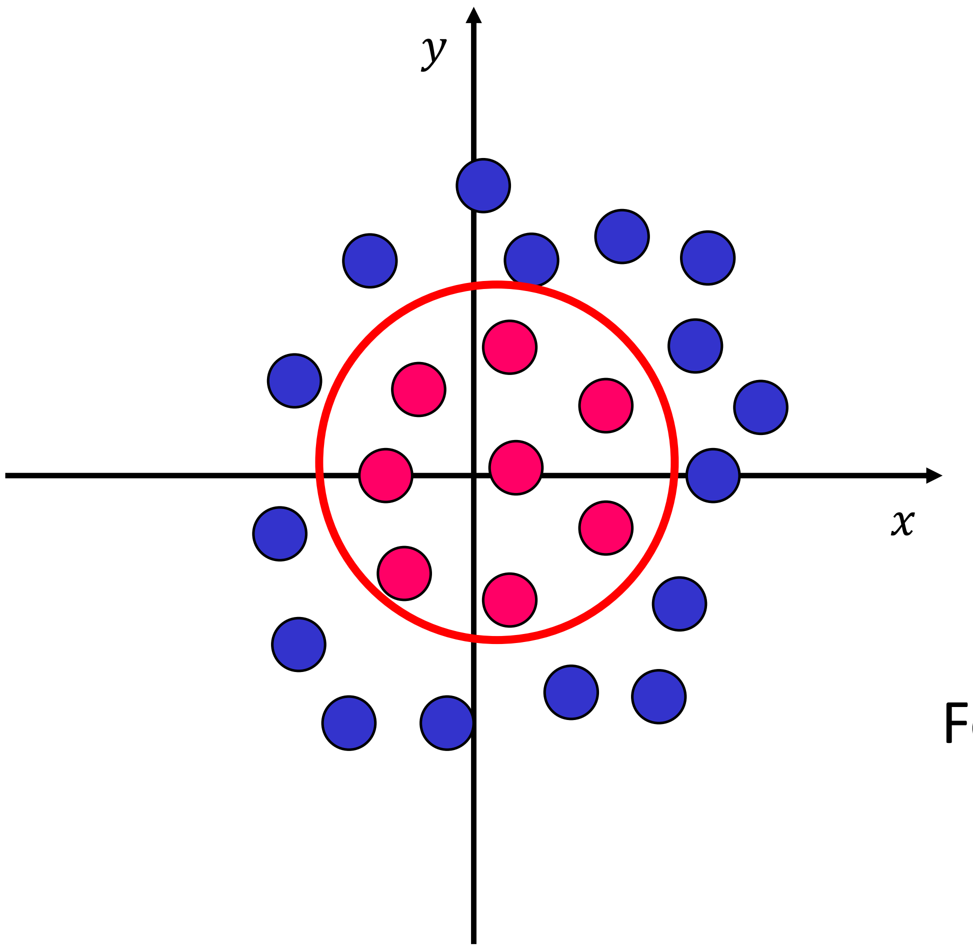
Feature transform

Feature space



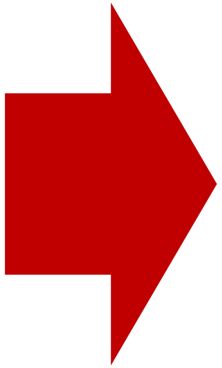
Solution: **Feature Transforms**

Original space



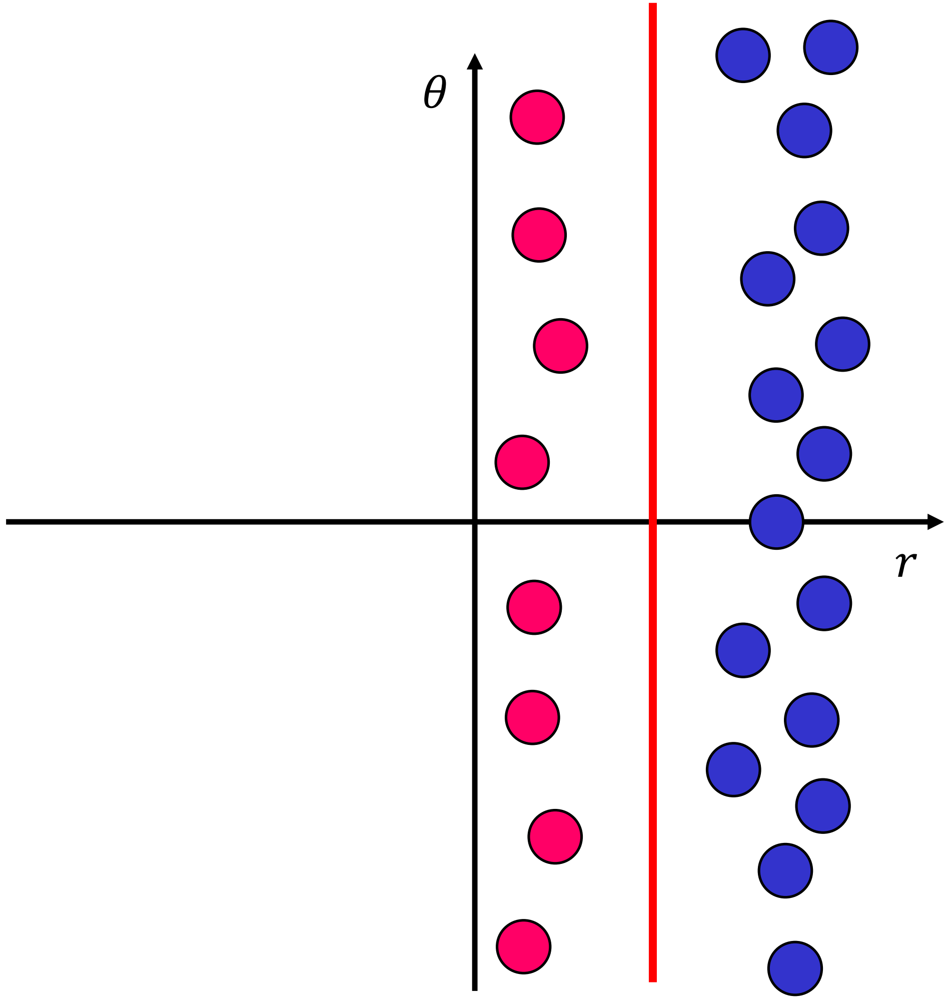
$$r = \sqrt{x^2 + y^2}$$

$$\theta = \tan^{-1}\left(\frac{y}{x}\right)$$



Feature transform

Feature space

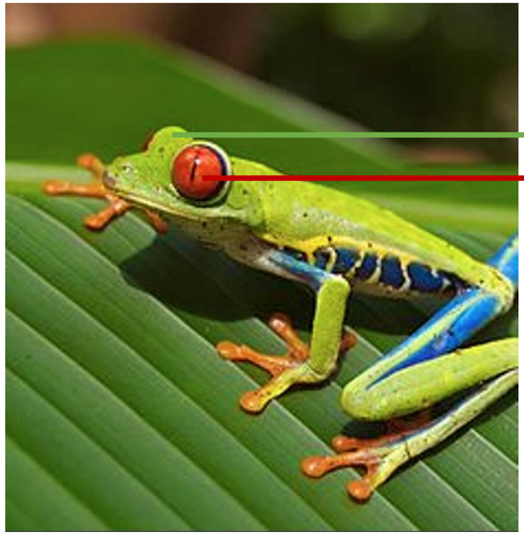


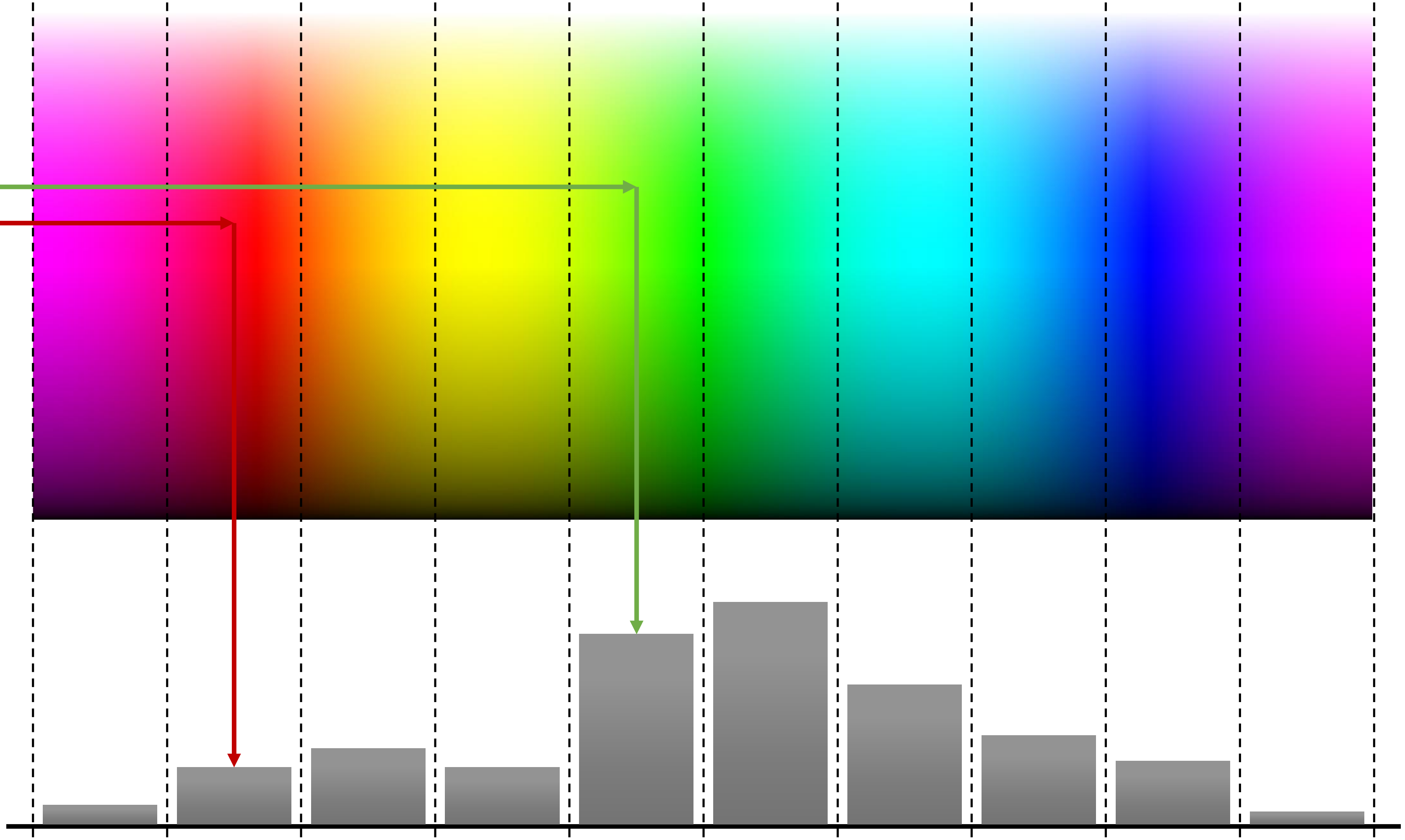
Nonlinear classifier in original space

Linear classifier in feature space

Image Features: **Color Histogram**

- A color histogram is a representation of **the distribution of colors**



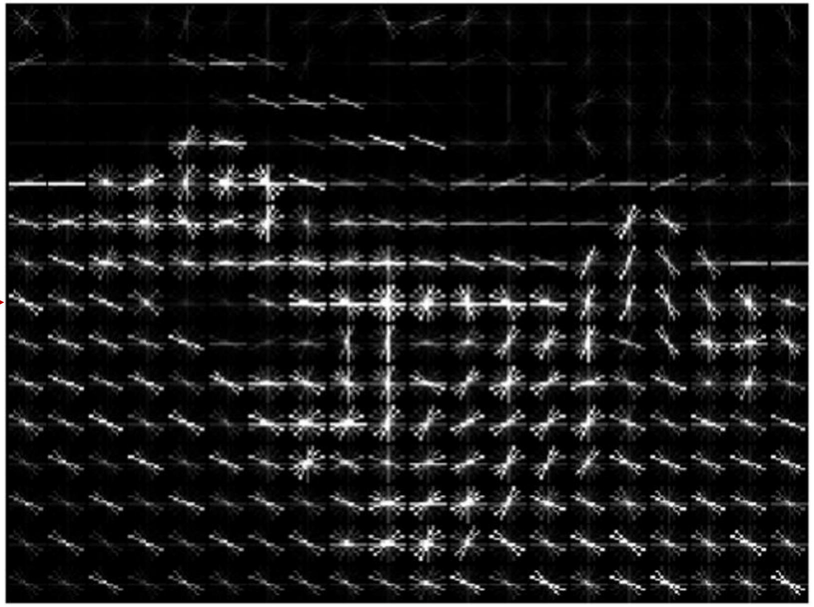


- **Limitation of color histogram:**
  Ignoring texture and spatial positions

Image Features: **Histogram of Oriented Gradients (HoG)**

- The histogram of oriented gradients (HOG) is a feature descriptor that **counts occurrences of gradient orientation** in localized portions of an image
  ① Compute edge direction and strength at each pixel
  ② Divide image into 8x8 regions
  ③ Compute a histogram of edge directions weighted by edge strength at each region

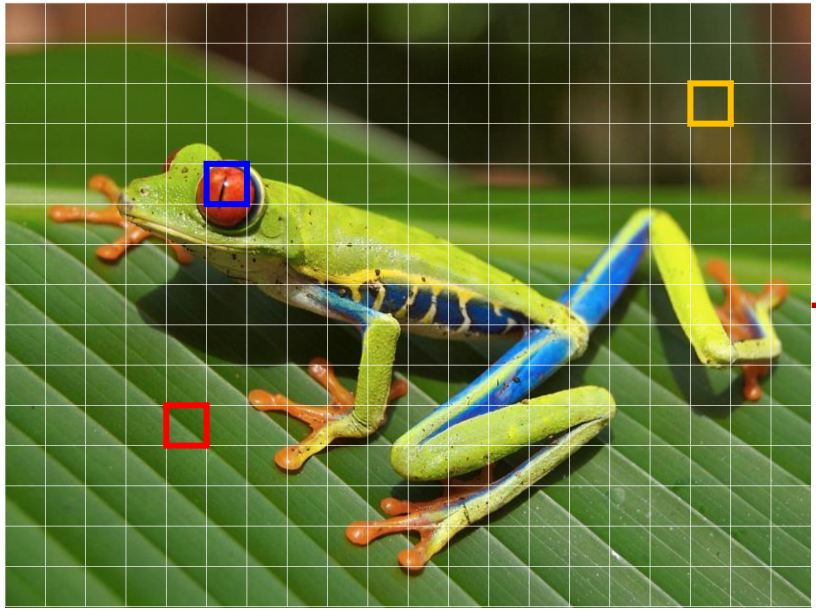
Original image



HoG features

Image Features: **Histogram of Oriented Gradients (HoG)**

- The histogram of oriented gradients (HOG) is a feature descriptor that **counts occurrences of gradient orientation** in localized portions of an image
  ① Compute edge direction and strength at each pixel
  ② Divide image into 8x8 regions
  ③ Compute a histogram of edge directions weighted by edge strength at each region

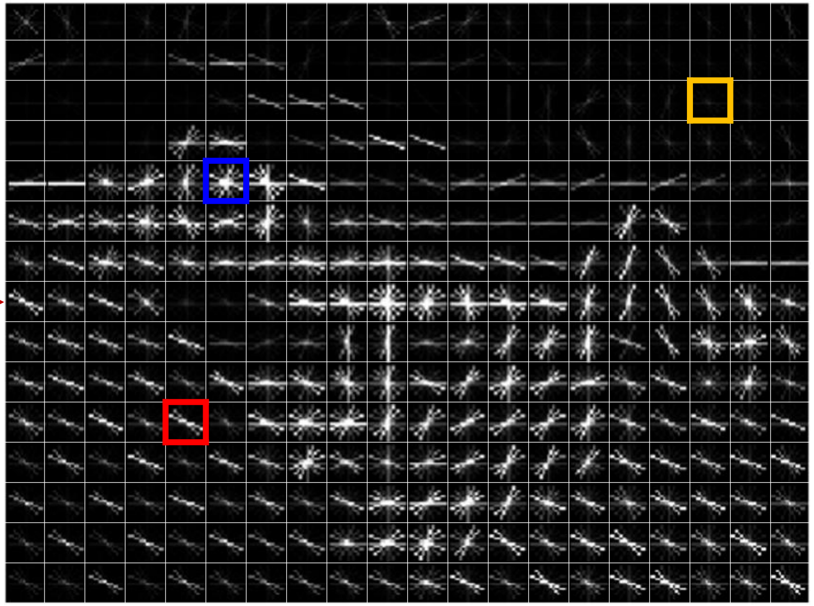


Original image

: Edges in all directions

: Strong diagonal edges

: Weak edges



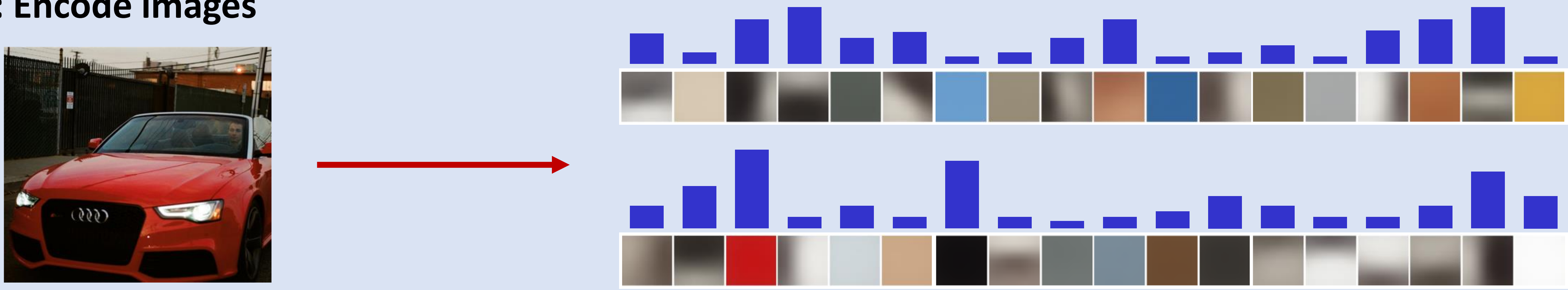
HoG features

Image Features: **Bag-of-Words (Data-driven Approach)**

- The bag-of-words model called **bag-of-visual-words** model can be applied to image classification, by treating image features **as words (local feature)**

Step 1: Build codebook

Step 2: Encode images



L. Fei Fei et al., A bayesian hierarchical model for learning natural scene categories, **CVPR 2005**

Image Feature Encoding in Computer Vision

- **For Image Feature Encoding:**

① Extract a bunch of different feature representations of input image

② Concatenate all together into one long high-dimensional feature vector that describes the input image

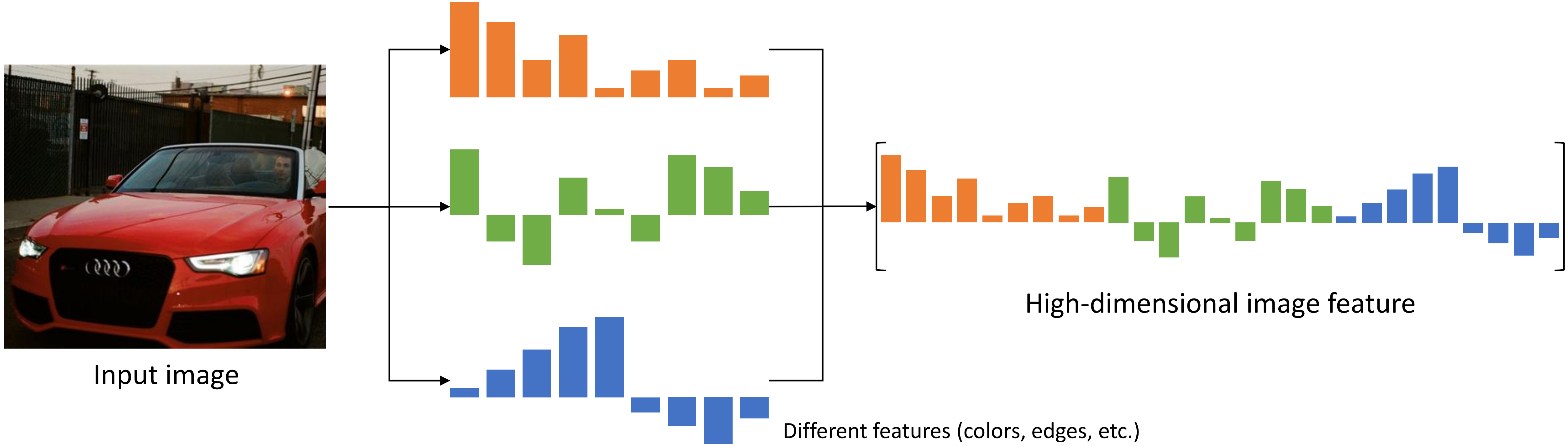
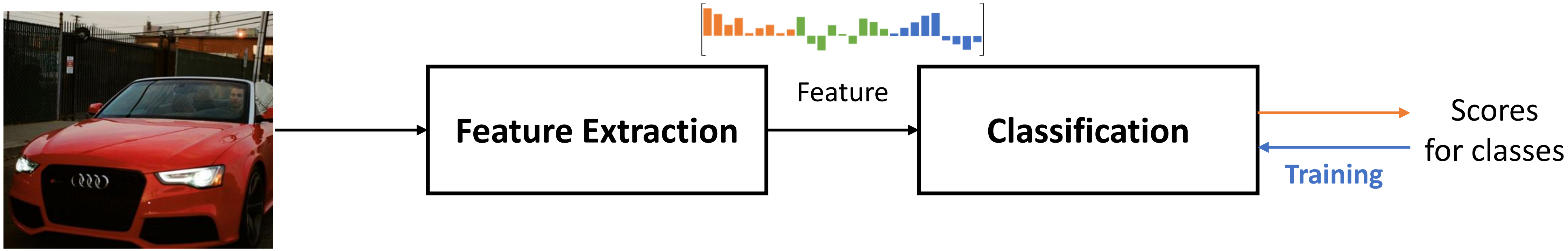
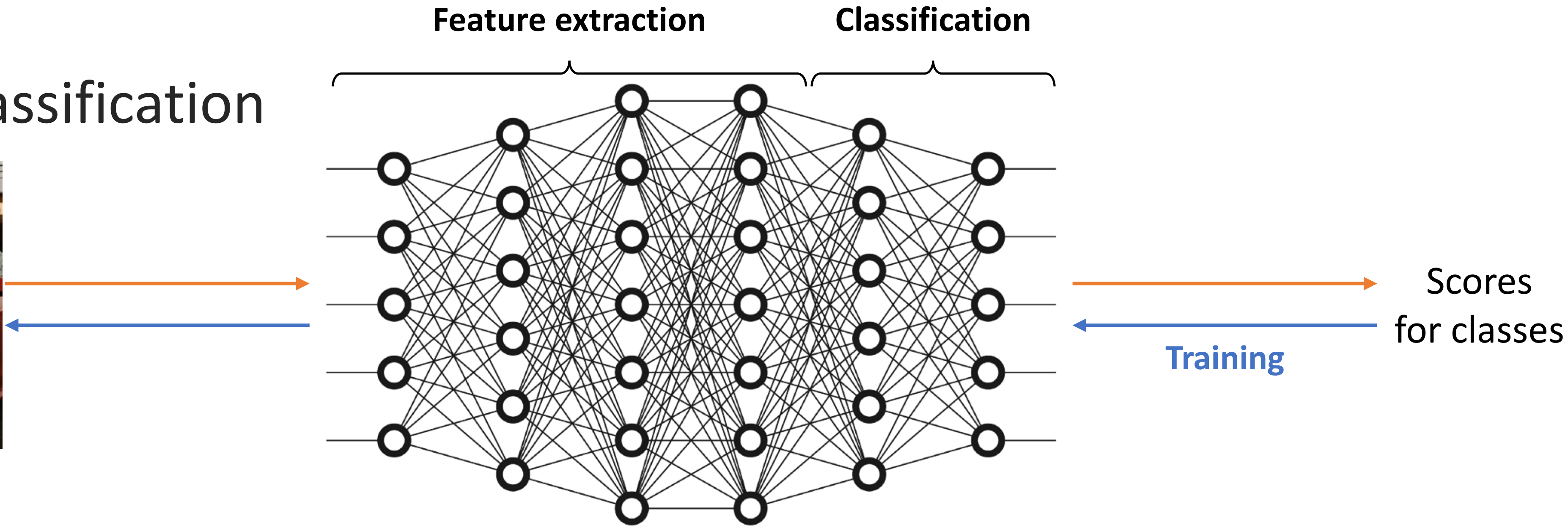


Image Classification Framework

- Conventional Image Classification



- DL-based Classification



Summary: **Image Classification**

- **The combination of feature transformation and linear classifier** allows **nonlinear** decision boundaries

- **Neural networks** can **learn both feature transform and classifier** in training. The neural networks are loosely inspired by biological neurons but be careful with analogies

Topics

- Classification
  - Linear Classifier with Images
  - Image Features for Robust Image Classification
  - Decision Tree

Classifiers

- Classification strategies: **parametric** and **nonparametric**

- **Parametric** classifiers are **model driven**

— The parameters of the model are learned from training examples

— New data points are classified by the learned model

- **Non-parametric** classifiers are **data driven**

— New data points are classified by comparing to the training examples directly

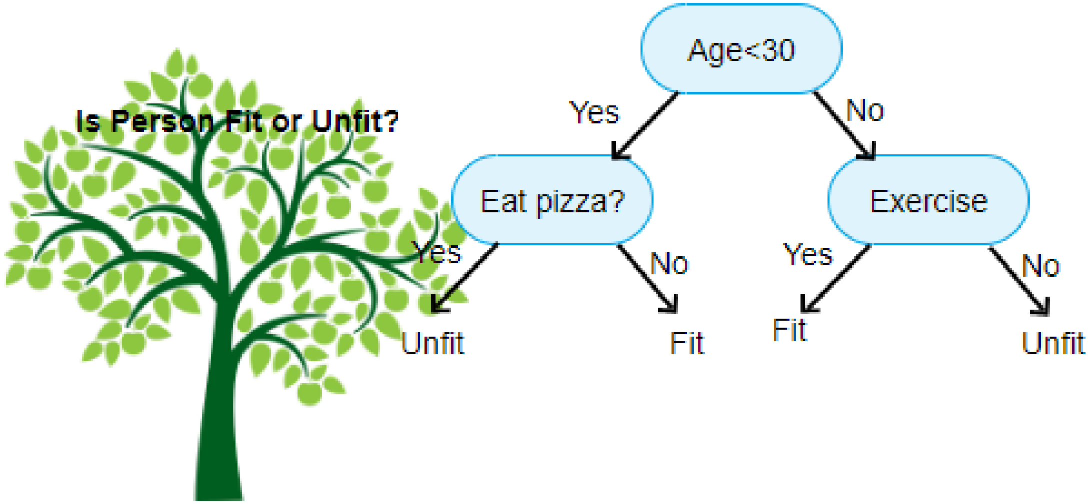
— *The data is the model*

Decision Tree

- A **decision tree** is a simple non-linear parametric classifier, consists of a tree in which each internal node is associated with a feature test

— A data point starts at the root and recursively proceeds to the child node determined by the feature test, until it reaches a leaf node

— The leaf node stores a class label or a probability distribution over class labels



Decision Tree

- Learning a decision tree from a training set involves selecting an efficient sequence of feature tests

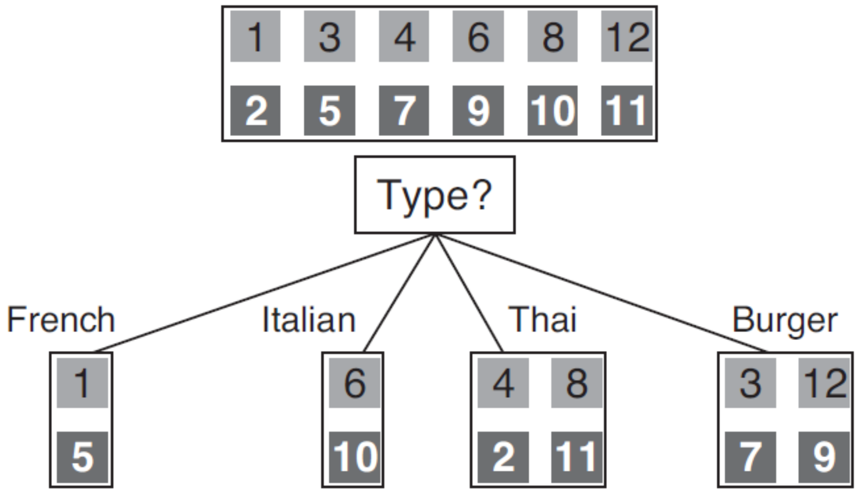
— Build a decision tree to decide whether to wait for a table at a restaurant

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

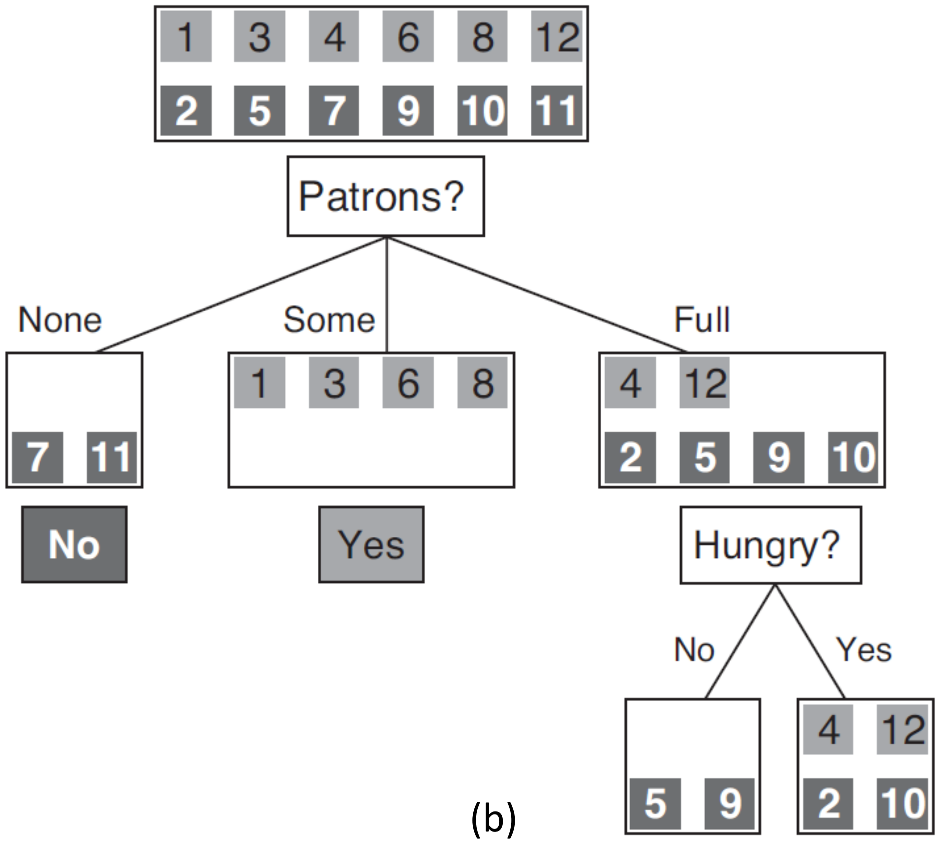
- **Alternate:** whether there is a suitable alternative restaurant nearby
- **Bar:** whether the restaurant has a comfortable bar area to wait in
- **Fri/Sat:** true on Fridays and Saturdays
- **Hungry:** whether we are hungry
- **Patrons:** how many people are in the restaurant
- **Price:** the restaurant's price range
- **Raining:** whether it is raining outside
- **Reservation:** whether we made a reservation
- **Type:** the kind of restaurant
- **WaitEstimate:** the wait estimated by the host

Decision Tree

- At each node we show the **positive (light boxes)** and **negative (dark boxes)** examples remaining



(a)
Splitting on *Type* brings us no nearer to distinguishing between positive and negative examples



(b)
Splitting on *Patrons* does a good job of separating positive and negative examples

Decision Tree: **Entropy**

- The **entropy** of a set S of data samples is defined as:

$$H(S) = -\sum_{c \in C} p(c) \log\big(p(c)\big)$$

  where C is the set of classes represented in S, and $p(c)$ is the empirical distribution of class c in S

- Entropy is highest when data samples are spread equally across all classes, and zero when all data samples are from the same class

Decision Tree: **Entropy**

- The **entropy** of a set S of data samples is defined as:

$$H(S) = -\sum_{c \in C} p(c) \log\big(p(c)\big)$$

where C is the set of classes represented in S, and $p(c)$ is the empirical distribution of class c in S

- **Example** of the entropy: The entropy of a fair coin flip is indeed 1 bit

$$H(Fair) = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1$$

Decision Tree: **Information Gain**

- In general, we try to select the feature test that maximizes the **information gain:**

$$IG(S, a) = H(S) - H(S|a)$$

where $H(S|a)$ is the conditional entropy of S given the value of attribute a

$$H(S|a) = \sum_{v \in vals(a)} \frac{|S_a(v)|}{|S|} H\big(S_a(v)\big)$$

where $S_a(v) = \{\mathbf{x} \in S \mid x_a = v\}$ is the set of training input of S for which attribute a is equal to v

Decision Tree: **Random Forest**

- A **random forest** is an ensemble of decision trees
  - Randomness is incorporated via training set sampling and/or generation of the candidate binary tests
  - The prediction of the random forest is obtained by averaging over all decision trees

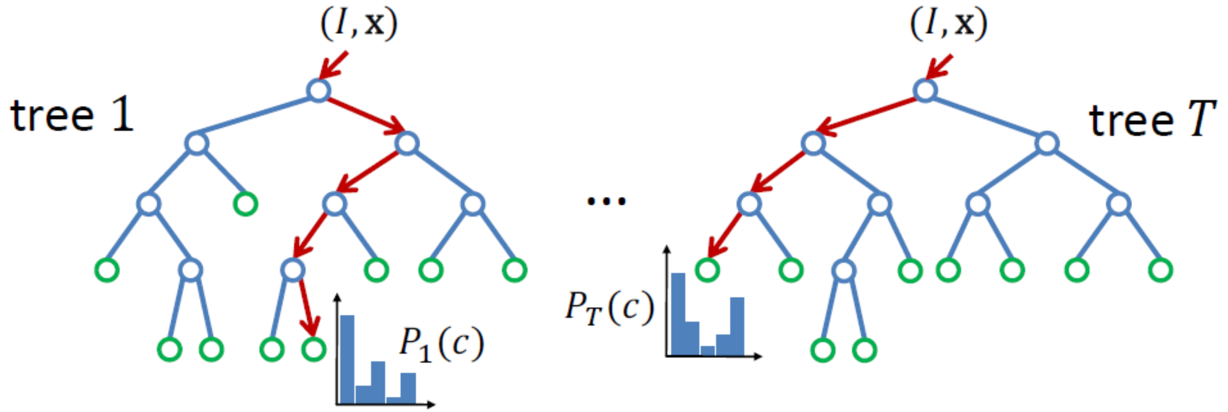


Figure 4. **Randomized Decision Forests.** A forest is an ensemble of trees. Each tree consists of split nodes (blue) and leaf nodes (green). The red arrows indicate the different paths that might be taken by different trees for a particular input.

J. Shotton et al., Real-Time Human Pose Recognition in Parts from Single Depth Images, **CVPR 2011**

Example of Random Forest: **Human Pose Recognition**

- **Kinect** allows users of Microsoft's Xbox 360 console to interact with games using natural body motions instead of a traditional handheld controller
- The pose (joint positions) of the user is predicted using a random forest trained on depth features

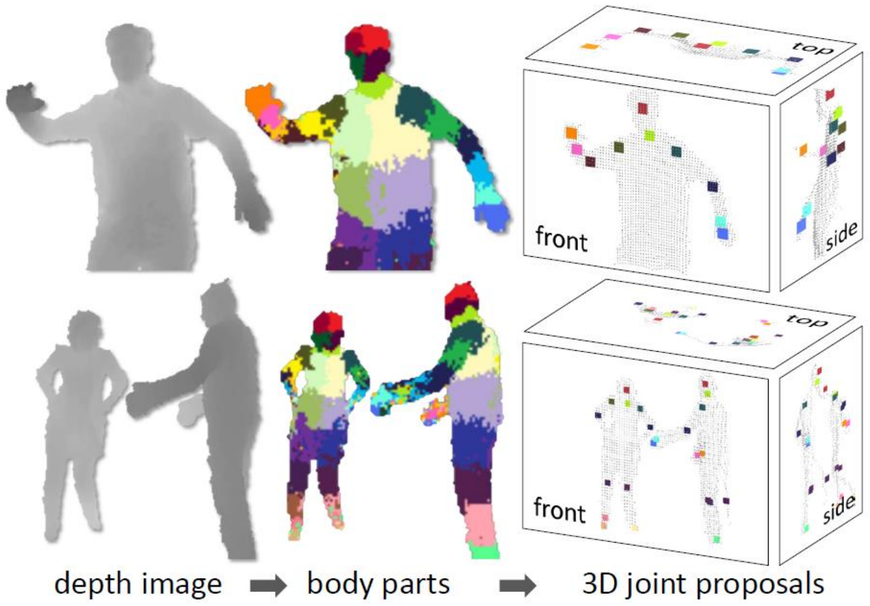


Figure 1. **Overview.** From an single input depth image, a per-pixel body part distribution is inferred. (Colors indicate the most likely part labels at each pixel, and correspond in the joint proposals).

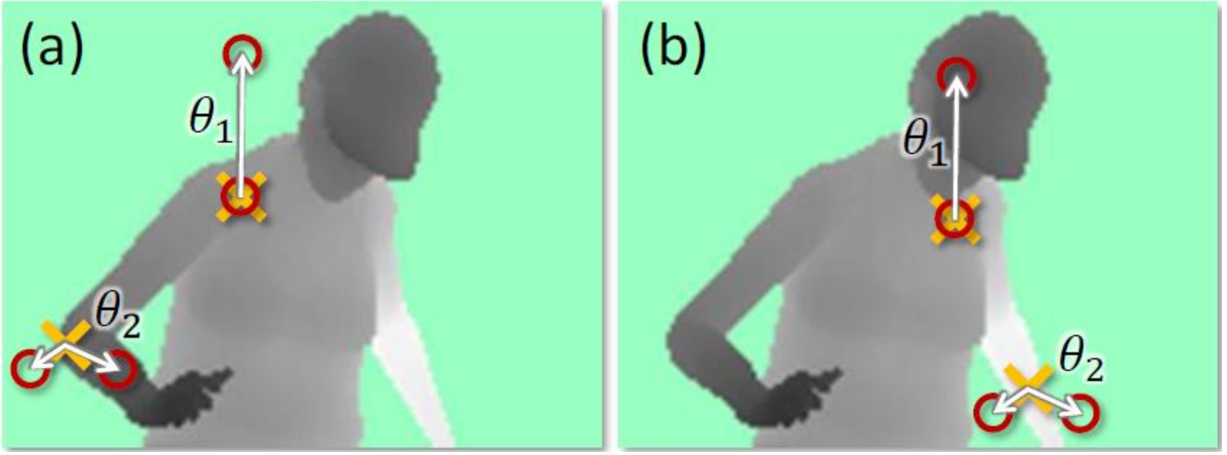
J. Shotton et al., Real-Time Human Pose Recognition in Parts from Single Depth Images, **CVPR 2011**

Example of Random Forest: **Human Pose Recognition**

- At a given pixel $\mathbf{x}$, the depth image features can be computed as:

$$f_\theta(I, \mathbf{x}) = d_I\left(\mathbf{x} + \frac{\mathbf{u}}{d_I(\mathbf{x})}\right) - d_I\left(\mathbf{x} + \frac{\mathbf{v}}{d_I(\mathbf{x})}\right)$$

where $d_I(\mathbf{x})$ is the depth at pixel $\mathbf{x}$ in image I, and parameters $\theta = (\mathbf{u}, \mathbf{v})$ describe offset $\mathbf{u}$ and $\mathbf{v}$



Depth image features: yellow crosses indicate the pixel $\mathbf{x}$ and the red circles indicate the offset pixels

J. Shotton et al., Real-Time Human Pose Recognition in Parts from Single Depth Images, **CVPR 2011**

Example of Random Forest: **Human Pose Recognition**

- An ensemble of T decision trees, each consisting of split and leaf nodes
  - Each split node consists of a feature f_θ and a threshold τ
  - To classify pixel $\mathbf{x}$ in image I, one starts at the root and repeatedly evaluates the feature f_θ, branching left or right according to the comparison to threshold τ
  - At the leaf node reached in tree t, a learned distribution $P(c|I, \mathbf{x})$ over body part labels is stored
  - The distributions are averaged together for all trees in the forest to give the final classification

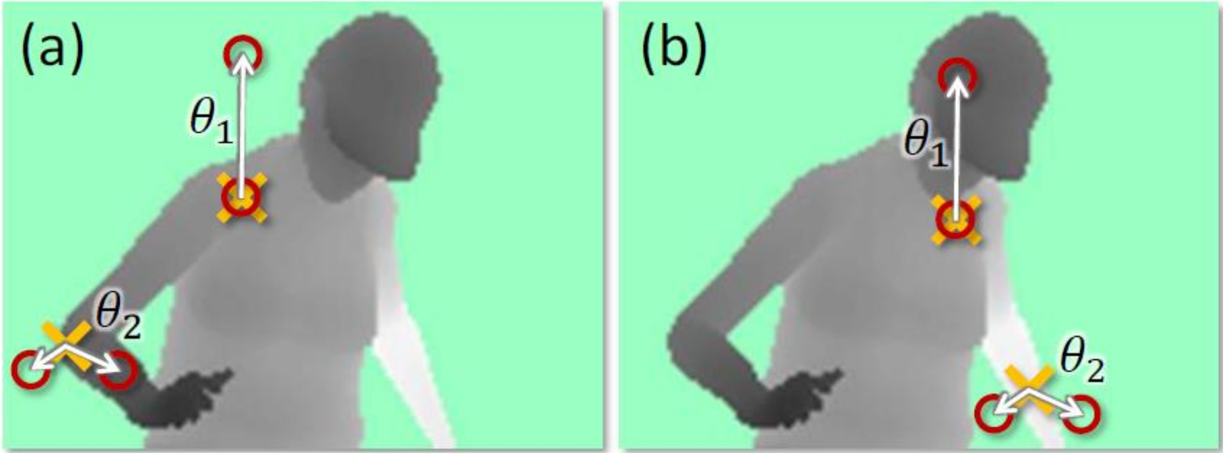
$$P(c|I, \mathbf{x}) = \frac{1}{T} \sum_{t=1}^{T} P_t(c|I, \mathbf{x})$$

J. Shotton et al., Real-Time Human Pose Recognition in Parts from Single Depth Images, **CVPR 2011**

Example of Random Forest: **Human Pose Recognition**

$\mathbf{u}$	$\mathbf{v}$	$\boldsymbol{\Theta_j}$	$\boldsymbol{IG}$
↙	⟶	0.5	0.3
↙	⟶	0.4	0.4
↙	⟶	-0.2	0.7
↙	↑	0.7	0.2
←	↑	-0.7	0.8
⟶	↖	0.45	0.1
⋮	⋮	⋮	⋮

$$f_\theta(I, \mathbf{x}) > \Theta_j$$

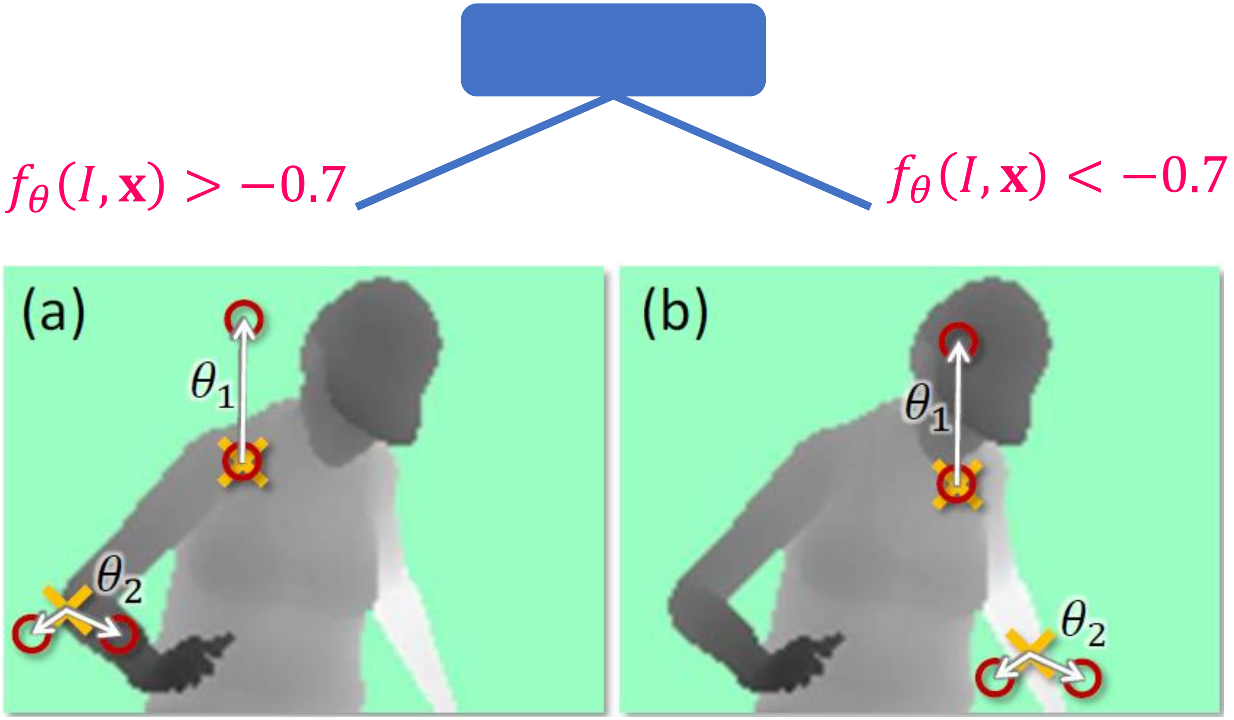


$$f_\theta(I, \mathbf{x}) = d_I\left(\mathbf{x} + \frac{\mathbf{u}}{d_I(\mathbf{x})}\right) - d_I\left(\mathbf{x} + \frac{\mathbf{v}}{d_I(\mathbf{x})}\right)$$

Example of Random Forest: **Human Pose Recognition**

$\mathbf{u}$	$\mathbf{v}$	Θ_j	IG
↙	⟶	0.5	0.3
↙	⟶	0.4	0.4
↙	⟶	-0.2	0.7
↙	↑	0.7	0.2
←	↑	**-0.7**	**0.8**
⟶	↖	0.45	0.1
⋮	⋮	⋮	⋮

$$f_\theta(I, \mathbf{x}) = d_I(\mathbf{x} + \leftarrow) - d_I(\mathbf{x} + \uparrow)$$



$$f_\theta(I, \mathbf{x}) = d_I\left(\mathbf{x} + \frac{\mathbf{u}}{d_I(\mathbf{x})}\right) - d_I\left(\mathbf{x} + \frac{\mathbf{v}}{d_I(\mathbf{x})}\right)$$

Example of Random Forest: **Human Pose Recognition**

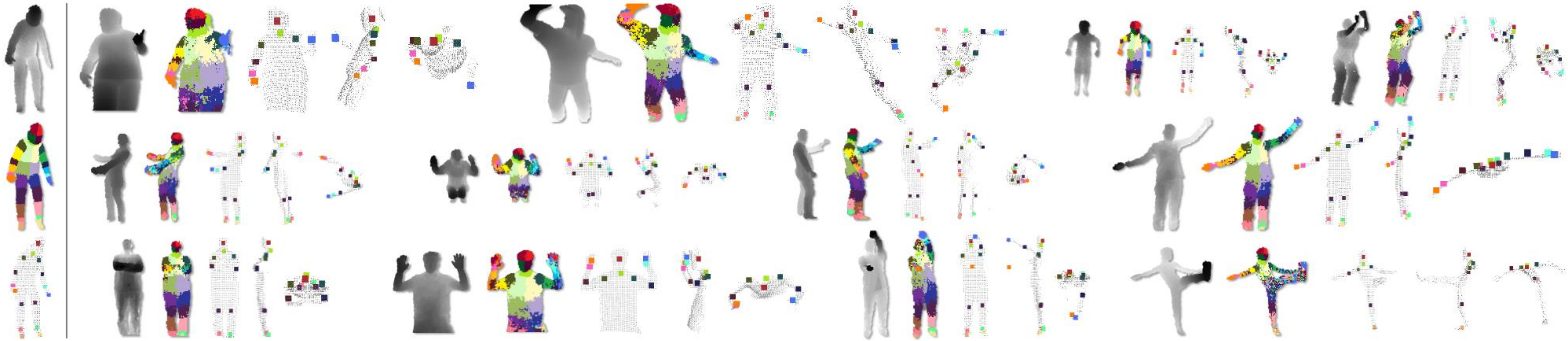


Figure 5. **Example inferences.** Synthetic (top row); real (middle); failure modes (bottom). Left column: ground truth for a neutral pose as a reference. In each example we see the depth image, the inferred most likely body part labels, and the joint proposals show as front, right, and top views (overlaid on a depth point cloud). Only the most confident proposal for each joint above a fixed, shared threshold is shown.

J. Shotton et al., Real-Time Human Pose Recognition in Parts from Single Depth Images, **CVPR 2011**

Combining **Classifiers**

- To obtain a better classifier, a simple approach is to train an ensemble of independent classifiers, and average their predictions

- **Boosting** is another approach:

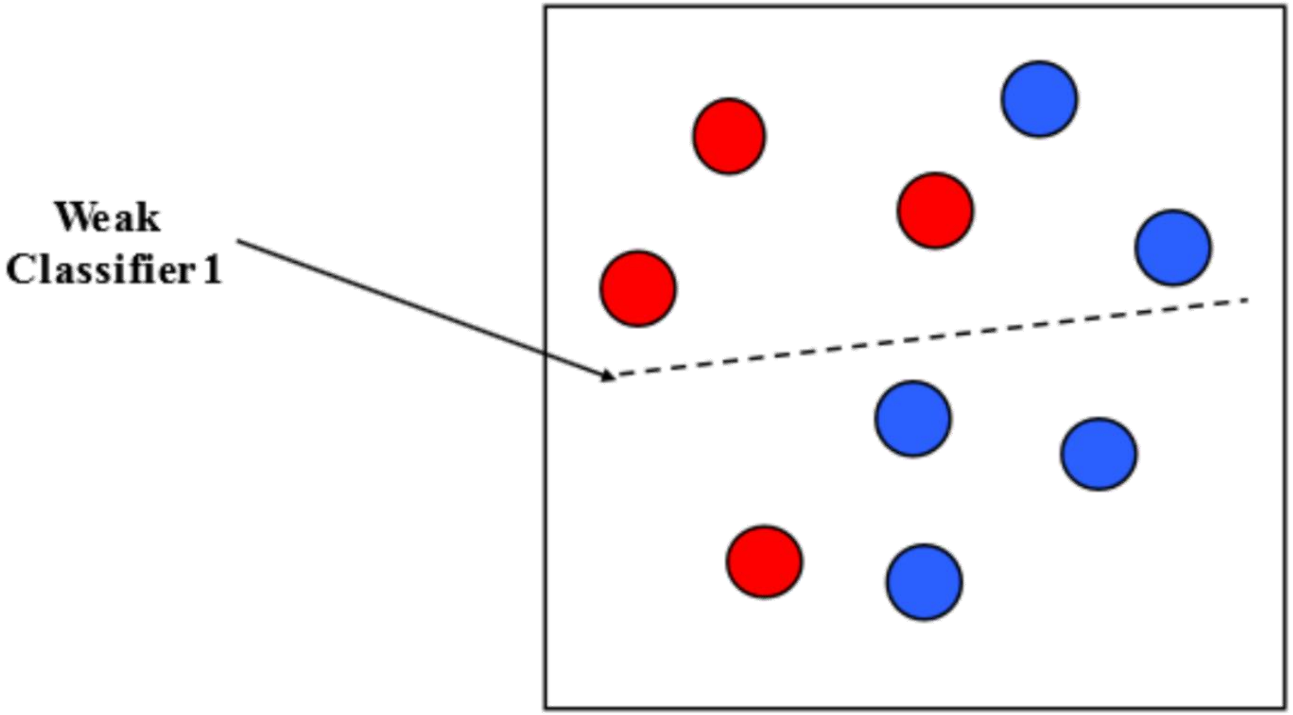
— Train an ensemble of classifiers sequentially

— Bias subsequent classifiers to correctly predict training examples that previous classifiers got wrong

— The final boosted classifier is a weighted combination of the individual classifiers

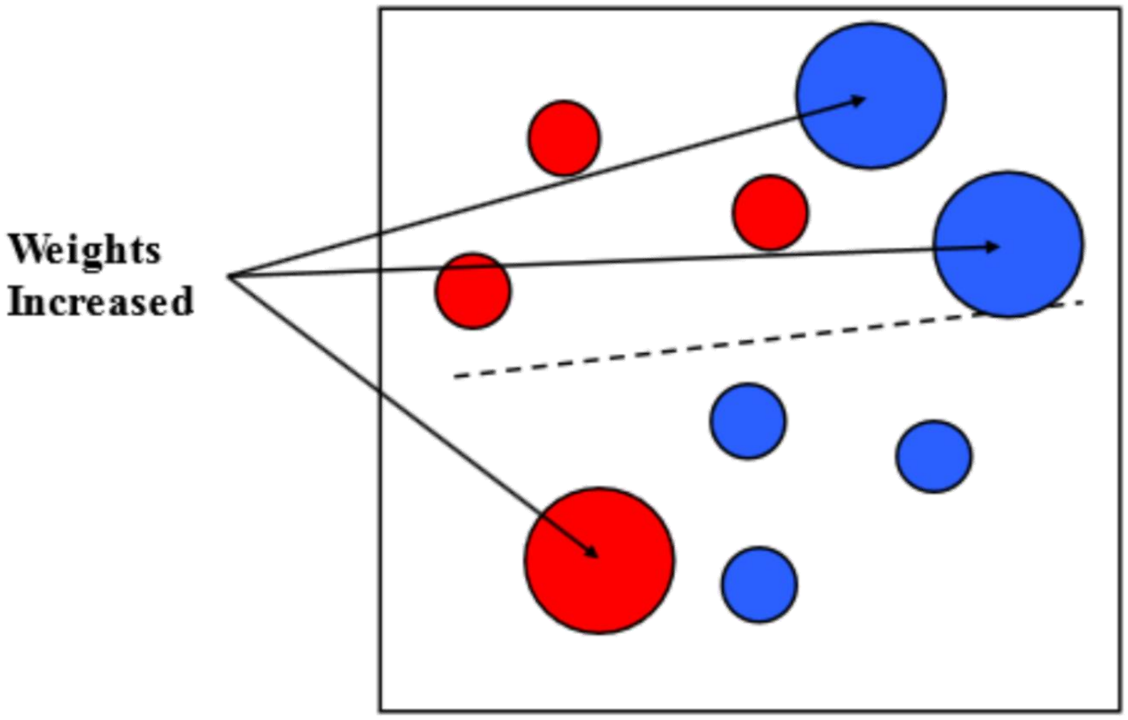
Combining Classifiers: **Boosting**

- In machine learning, **boosting** is an ensemble meta-algorithm for primarily reducing bias, and also variance in supervised learning



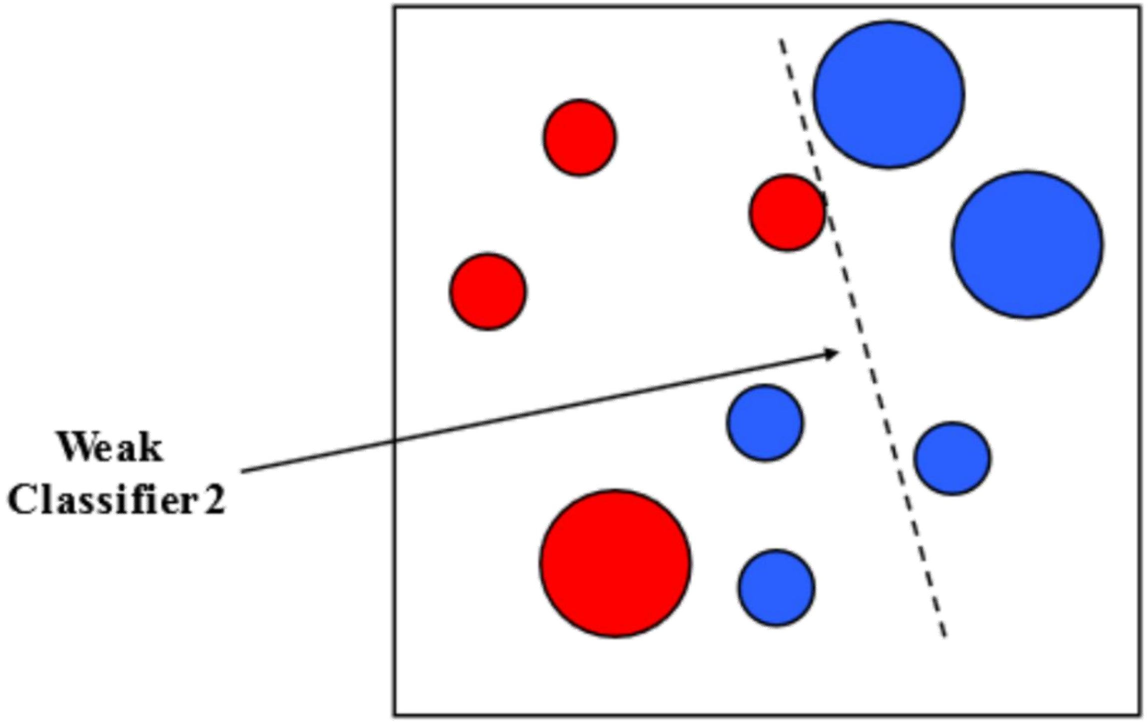
Combining Classifiers: **Boosting**

- In machine learning, **boosting** is an ensemble meta-algorithm for primarily reducing bias, and also variance in supervised learning



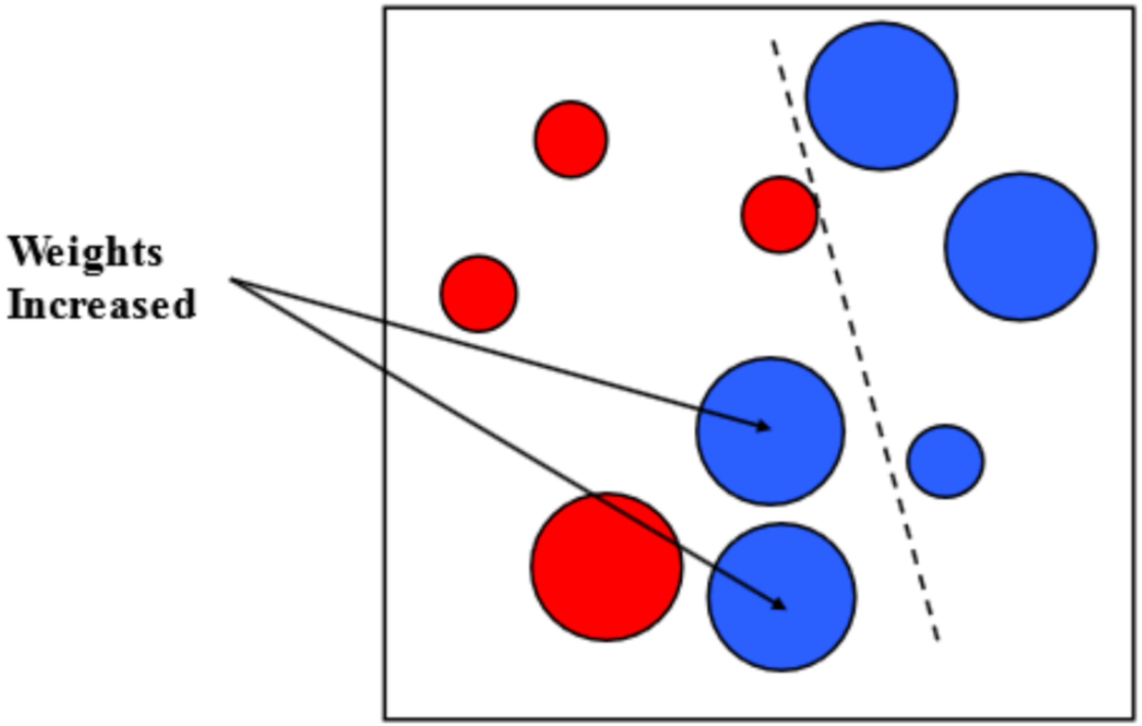
Combining Classifiers: **Boosting**

- In machine learning, **boosting** is an ensemble meta-algorithm for primarily reducing bias, and also variance in supervised learning



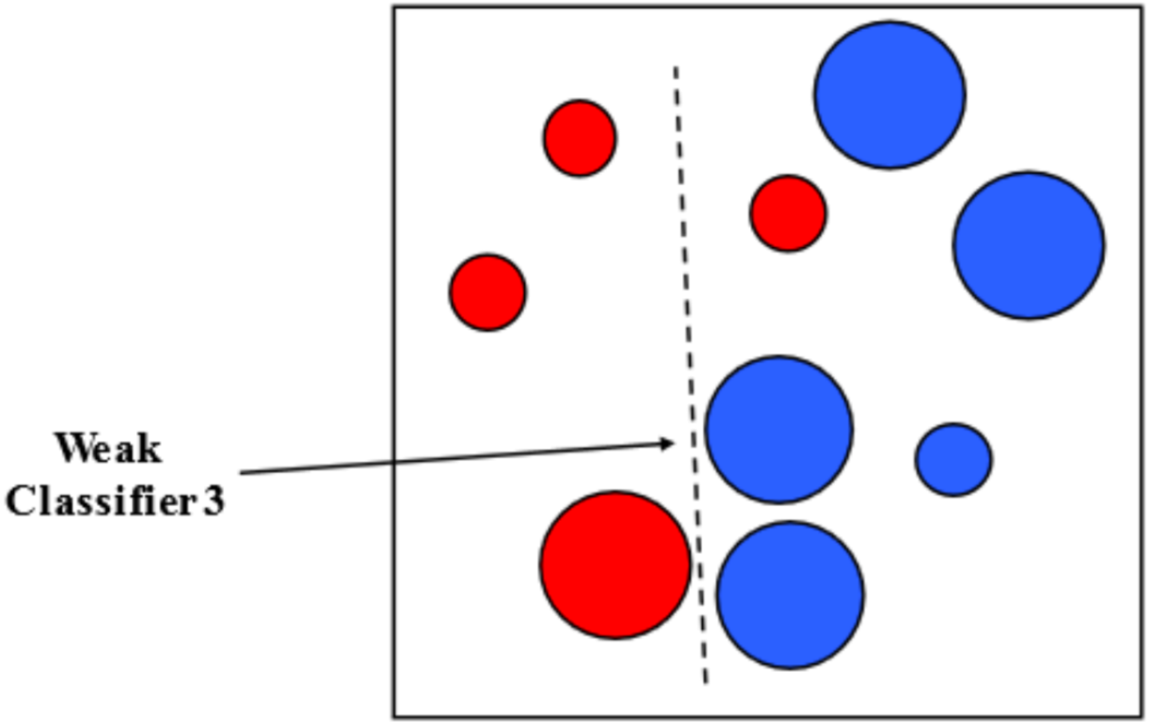
Combining Classifiers: **Boosting**

- In machine learning, **boosting** is an ensemble meta-algorithm for primarily reducing bias, and also variance in supervised learning



Combining Classifiers: **Boosting**

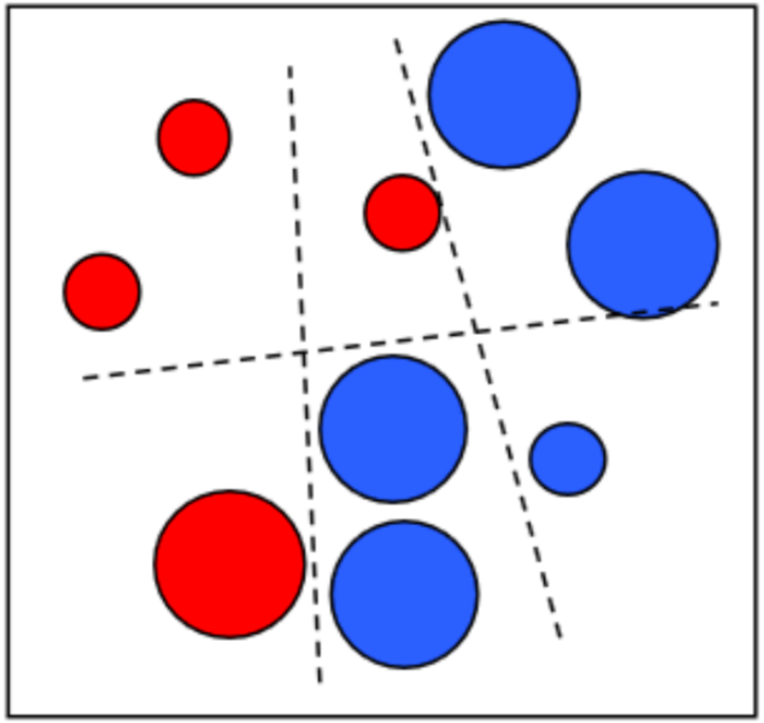
- In machine learning, **boosting** is an ensemble meta-algorithm for primarily reducing bias, and also variance in supervised learning



Combining Classifiers: **Boosting**

- In machine learning, **boosting** is an ensemble meta-algorithm for primarily reducing bias, and also variance in supervised learning

Final classifier is a combination of weak classifiers



Summary

- Classification
  - **The linear classifier, f, with weight and bias terms** is defined to map the pixel values of an image to confidence scores for each class
  - However, it is difficult to classify the data, which are nonlinearly distributed
  - To overcome the limitation, **feature encoding** is proposed such as HoG, BoW, *etc*.

  - **Parametric** classifiers are model driven
    - The parameters of the model are learned from training examples
    - New data points are classified by the learned model
  - **Non-parametric** classifiers are data driven
    - New data points are classified by comparing to the training examples directly
    - The data is the model

- **Combining** Classifiers
  - To obtain a better classifier, a simple approach is to train an ensemble of independent classifiers, and average their predictions