Robot Vision

TTK4255

Lecture 13 – Detection and Recognition

Annette Stahl

(Annette.Stahl@ntnu.no)

Department of Engineering Cybernetics – ITK

NTNU, Trondheim

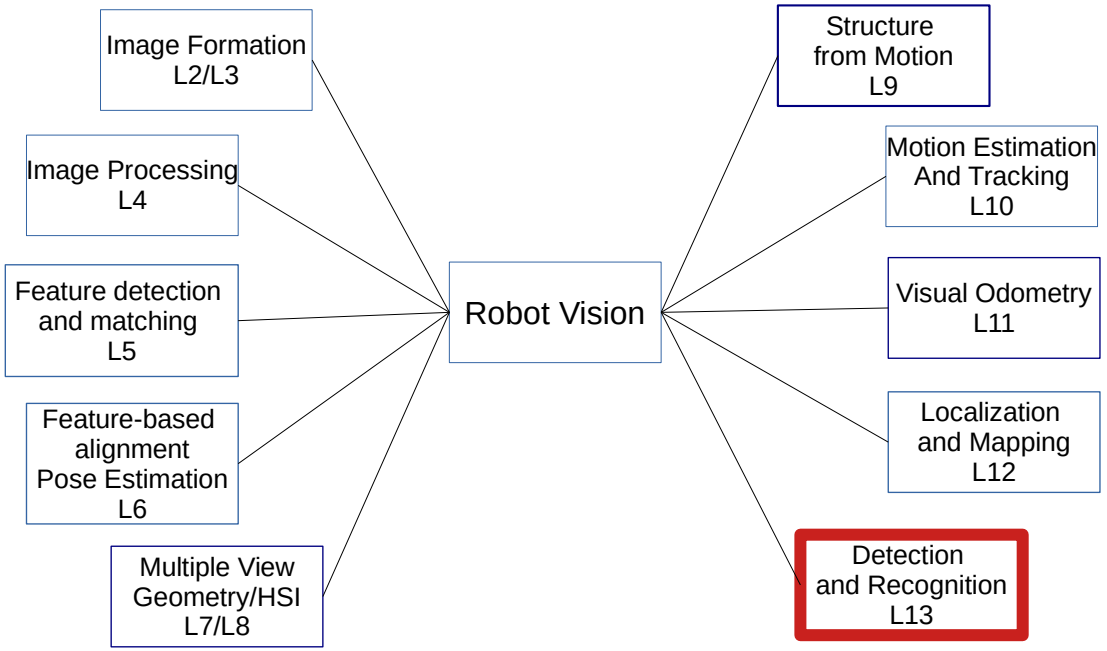
Spring Semester

30. March 2020

Lecture 13 – Detection and Recognition

Annette Stahl (Annette.Stahl@ntnu.no)

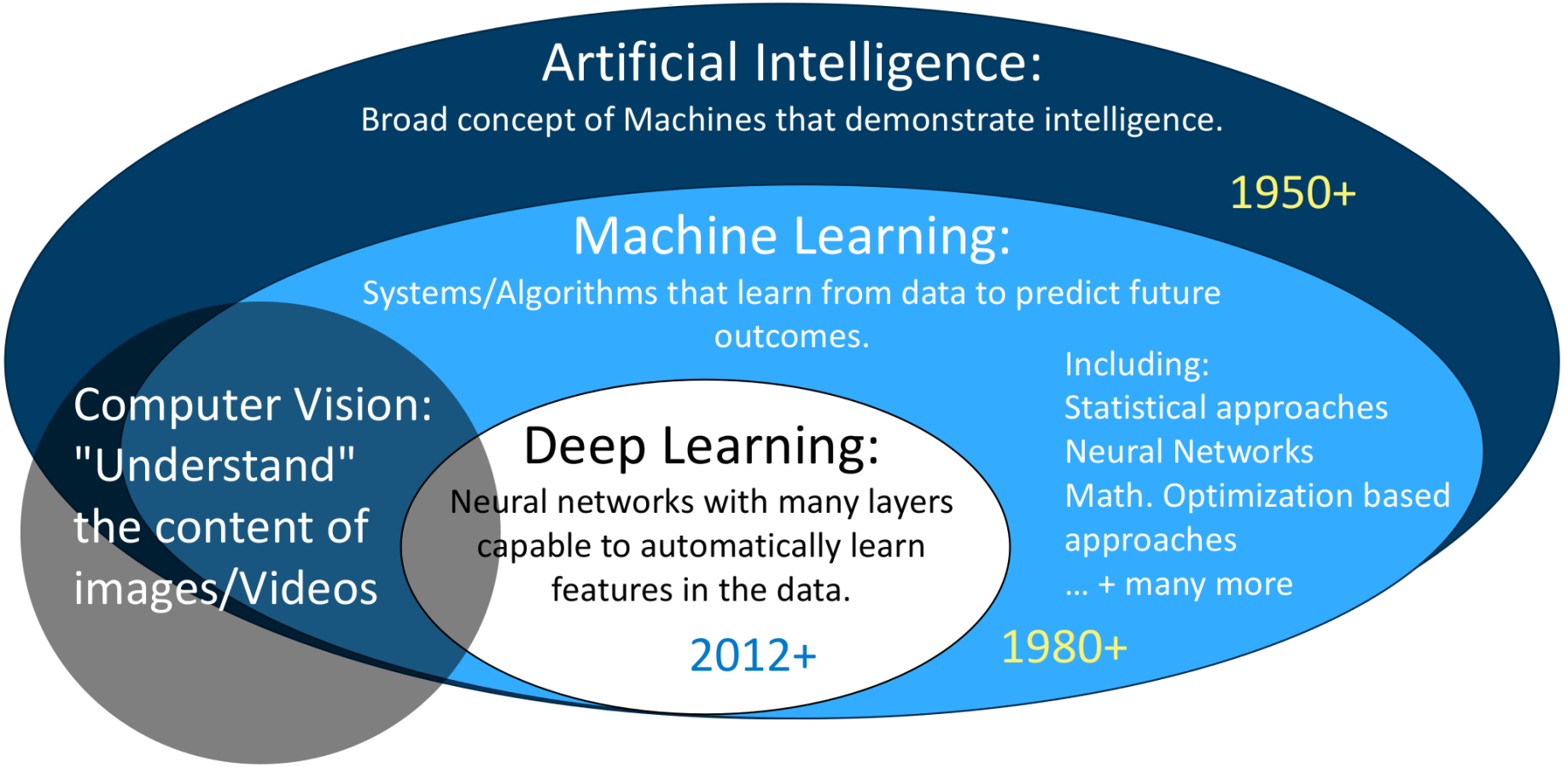
Simen Haugo (Simen.Haugo@ntnu.no)



Outline of the thirteenth lecture:

- Recognition
- Face Detection and Boosting
- Support Vector Machines
- Clustering: K-means, Mean-shift
- Deep Learning

Detection and Recognition



Courtesy: Christian Schellewald, 2020

Object Recognition

When we talk about object recognition, we talk about image content.

- This can mean verification: is this a street lamp?
- This can mean detection: are there people present? If so, where?
- This can mean identification: is this Potala Palace?
- This can mean categorization.
- It can mean segmentation: labeling all pixels with category label.
- It can mean attribution: labeling objects with attributes (flat, hairy, circular, etc).
- It can mean various things, that have something to do with associating meaning to image content.

Andrew D. Bagdanov. Object Recognition in images. CVPR workshop 2016

Object Recognition

When we talk about object recognition, we talk about image content.

- This can mean verification: is this a street lamp?
- This can mean detection: are there people present? If so, where?
- This can mean identification: is this Potala Palace?
- This can mean categorization.
- It can mean segmentation: labeling all pixels with category label.
- It can mean attribution: labeling objects with attributes (flat, hairy, circular, etc).
- It can mean various things, that have something to do with associating meaning to image content.

Andrew D. Bagdanov. Object Recognition in images. CVPR workshop 2016

Object Recognition

When we talk about object recognition, we talk about image content.

- This can mean verification: is this a street lamp?
- This can mean detection: are there people present? If so, where?
- This can mean identification: is this Potala Palace?
- This can mean categorization.
- It can mean segmentation: labeling all pixels with category label.
- It can mean attribution: labeling objects with attributes (flat, hairy, circular, etc).
- It can mean various things, that have something to do with associating meaning to image content.

Andrew D. Bagdanov. Object Recognition in images. CVPR workshop 2016

Object Recognition

When we talk about object recognition, we talk about image content.

- This can mean verification: is this a street lamp?
- This can mean detection: are there people present? If so, where?
- This can mean identification: is this Potala Palace?
- This can mean categorization.
- It can mean segmentation: labeling all pixels with category label.
- It can mean attribution: labeling objects with attributes (flat, hairy, circular, etc).
- It can mean various things, that have something to do with associating meaning to image content.

Andrew D. Bagdanov. Object Recognition in images. CVPR workshop 2016

Object Recognition

When we talk about object recognition, we talk about image content.

- This can mean verification: is this a street lamp?
- This can mean detection: are there people present? If so, where?
- This can mean identification: is this Potala Palace?
- This can mean categorization.
- It can mean segmentation: labeling all pixels with category label.
- It can mean attribution: labeling objects with attributes (flat, hairy, circular, etc).
- It can mean various things, that have something to do with associating meaning to image content.

Andrew D. Bagdanov. Object Recognition in images. CVPR workshop 2016

Recognition

- Face detectors are some of the more successful examples of real-time recognition
- In principle, we could apply a face recognition algorithm at every pixel and scale (Moghaddam and Pentland 1997) but such a process would be too slow in practice.
- A wide variety of **fast face detection algorithms** have been developed and can be classified as
  - **Feature-based techniques** attempt to find the locations of distinctive image features such as the eyes,nose, and mouth, and then verify whether these features are in a plausible geometrical arrangement.
  - **Template-based approaches**, such as active appearance models (AAMs) can deal with a wide range of pose and expression variability. Typically, they require good initialization near a real face and are therefore not suitable as fast face detectors.
  - **Appearance-based approaches** scan over small overlapping rectangular patches of the image searching for likely face candidates, which can then be refined using a cascade of more expensive but selective detection algorithms. In order to deal with scale variation, the image is usually converted into a sub-octave pyramid and a separate scan is performed on each level. Most appearance-based approaches today rely heavily on training classifiers using sets of labeled face and non-face patches.

Szeliski Ch. 14.1.1

Face Detection

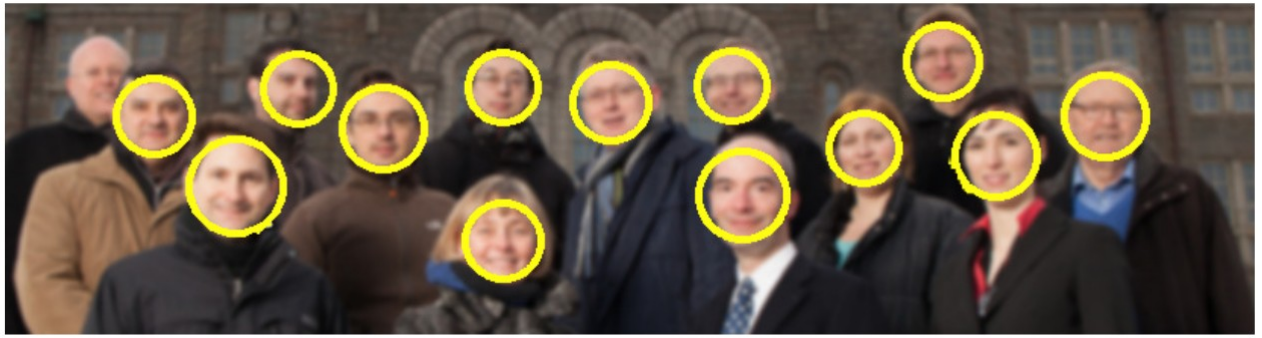
One of the first **real-time object detection framework**s was proposed by Paul Viola and Michael Jones in 2001 and was applied to the **detection of (frontal) faces**.

(Note: detection 6 = recognition)

→ OpenCV implementation of a variant of that framework

Key ingredients:

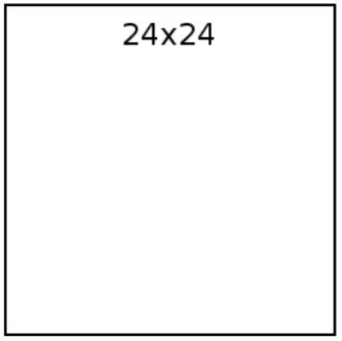
- Simple features (→ fast to compute)
- Selection of a small number of important features
- Cascade classification (complex processing only for promising regions)



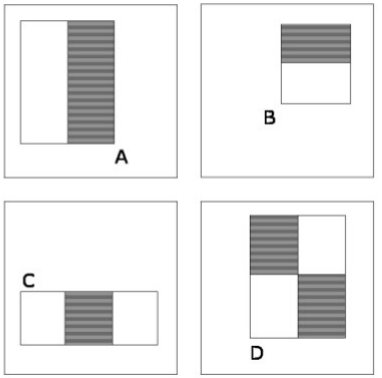
Features (Face Detection)

Simple features based on mean pixel-values in rectangular areas were used.

Base detector window



Example features



Single feature: The gray-value sum within the gray-shaded rectangle is subtracted from the sum within the white rectangle.

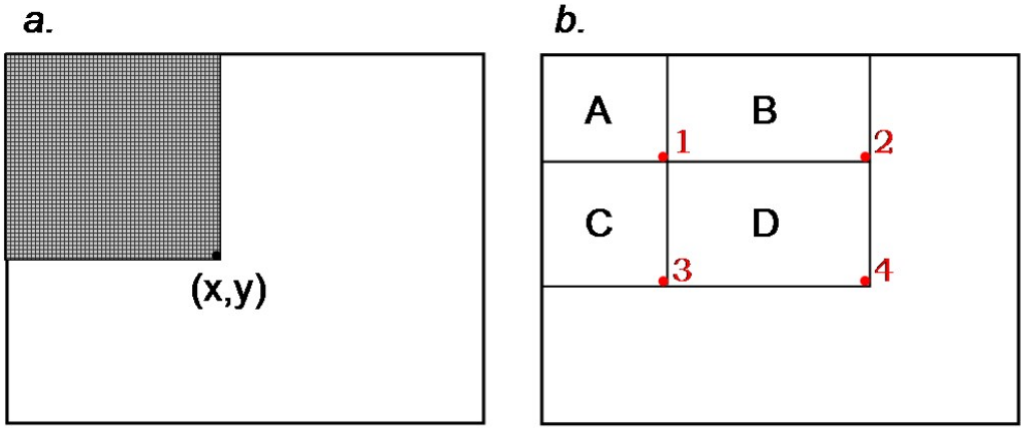
→ many possible features. (for a 24x24 pixel wide base window more than 180000).

For an analysis at many different positions and scales **many evaluations** of the sum of pixel values in rectangular areas must be computed.

→ Efficient do-able with an image representation known as **integral image**.

Integral Image

The so called integral image representation allows the quick computation of the sum (→mean) of any rectangular area in an image.



- The integral image at pixel (x,y) contains the sum of all pixel-values above and to the left of (x,y), inclusive. (= sum of gray-values in the shaded region)

  → The integral image can be computed in one pass over the original image

- The sum within rectangle D can be computed (in constant time) with just four array references:

  P4-P2-P3+P1

- Allows for a simple scaling of the feature detector

Weak Classifier

Extreme simple (sum/subtraction of pixel values in rectangular areas.

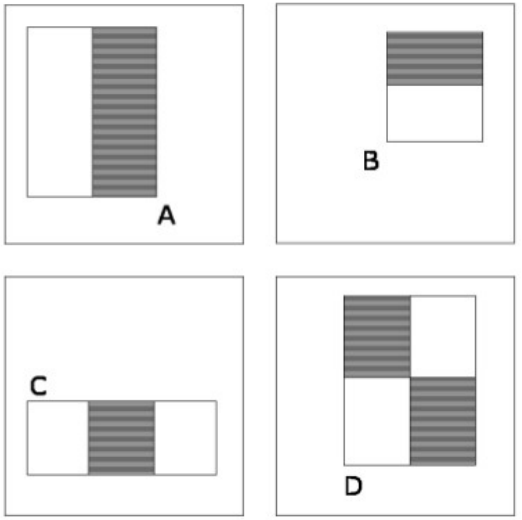
Efficient to compute:

→ Integral Image)

→ many possible features. (for a 24x24 pixel wide base window more than 180000).

Still impractical/impossible to evaluate all features at running time!

=> “Build a classifier using only a very small subset of features”



Feature Selection

Challenge: Find a very small number out of all possible features (180000) which can be combined to form an effective classifier.

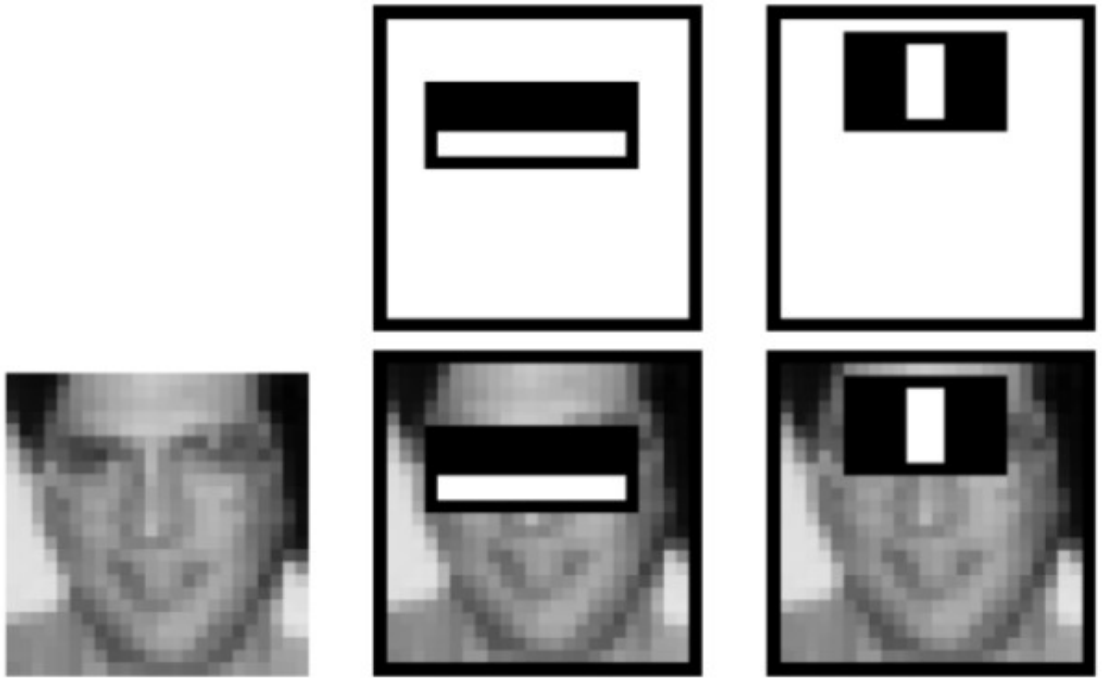
Basic Idea: Select the rectangle features which best separates the positive and negative training images.

Boosting: AdaBoost is an algorithm that allows to select one additional feature (=classifier) in each iteration to build a classifier that combines (by weighting) the results of the simple classifiers.

Positive training examples

The first two selected features

First two features selected



→ First classifier in cascade. “Removes 40% non-faces, keeps all real faces as candidates”

Classifier at later stages use more features ...

Training of the Classifier

Labeled Data:

- Faces normalized (scale, translation)
- All “frontal”

Large Variation

- Pose
- Illumination
- Many individuals

→ Much more non-faces !!!

AdaBoost (Feature Selection)

Given a set of weak classifiers:

$$h_j(x) \in \{+1, -1\}$$

→ none much better than random

Boosting iteratively adds and linearly combines weak learners

$$C(x) = \Theta\left(\sum_t h_t(x) + b\right)$$

→ Training error converges to 0

→ Adapt weights in training set (when misclassified by previous weak learner one increases weight)

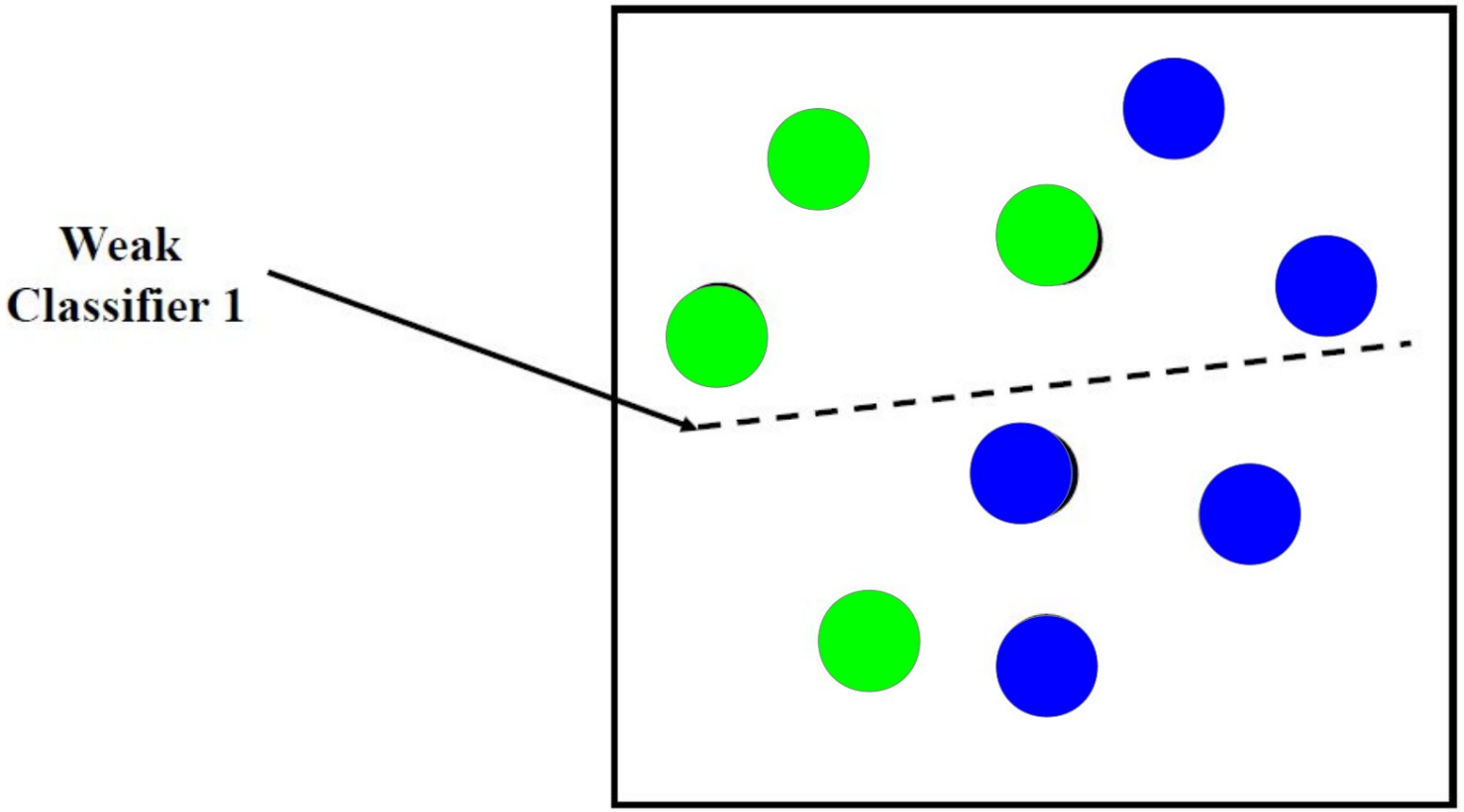
”AdaBoost is used for both to select a small set of features and train the classifier”

[AdaBoost: Freund, Schapire, 1995]

Idea of Boosting:

Select the 1st weak classifier that separates the 2 classes best.

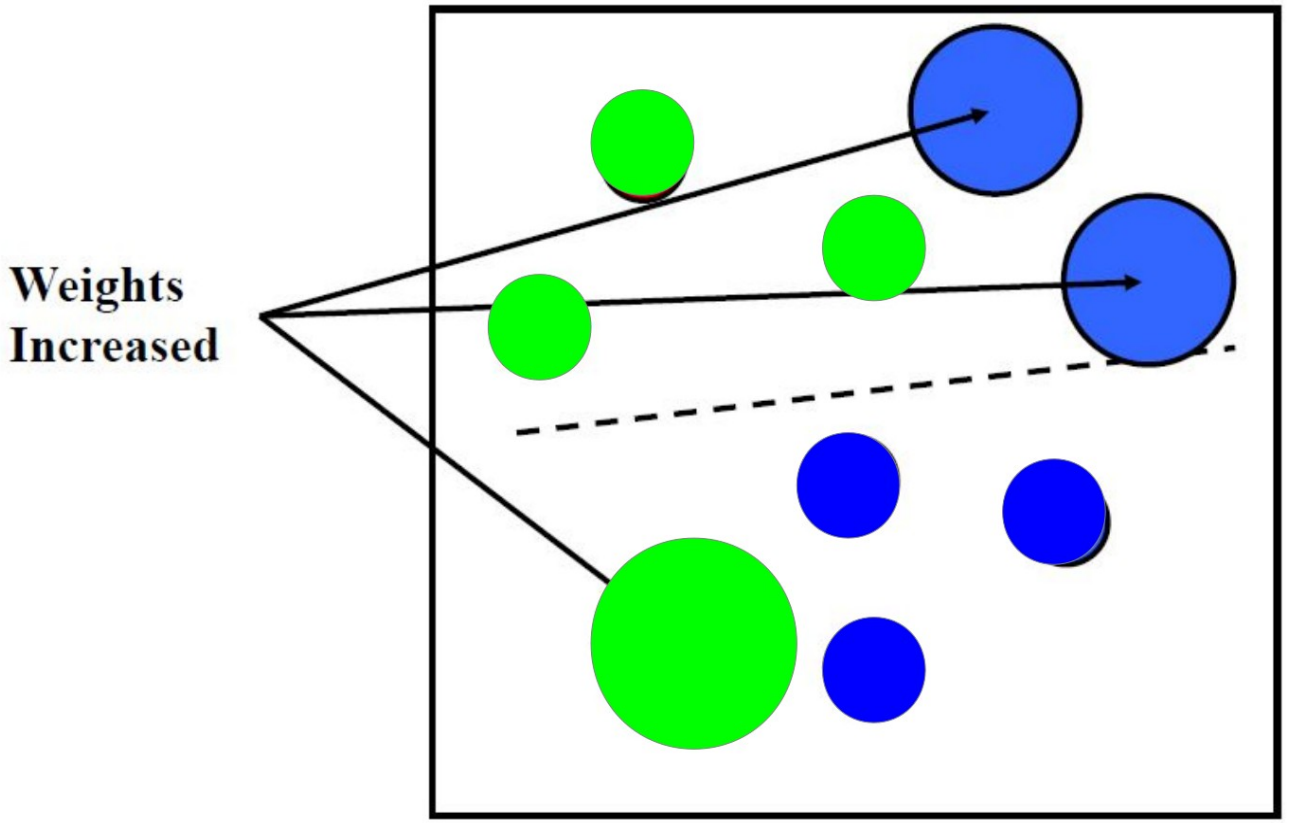
(For faces this needs to vary the feature [location+size] and threshold and is computationally very expensive. [We talk about days/weeks of training.])



Idea of Boosting:

Make misclassified examples more important (increase weights).

(Idea: The additional selected weak classifier, should perform better on this training examples.)

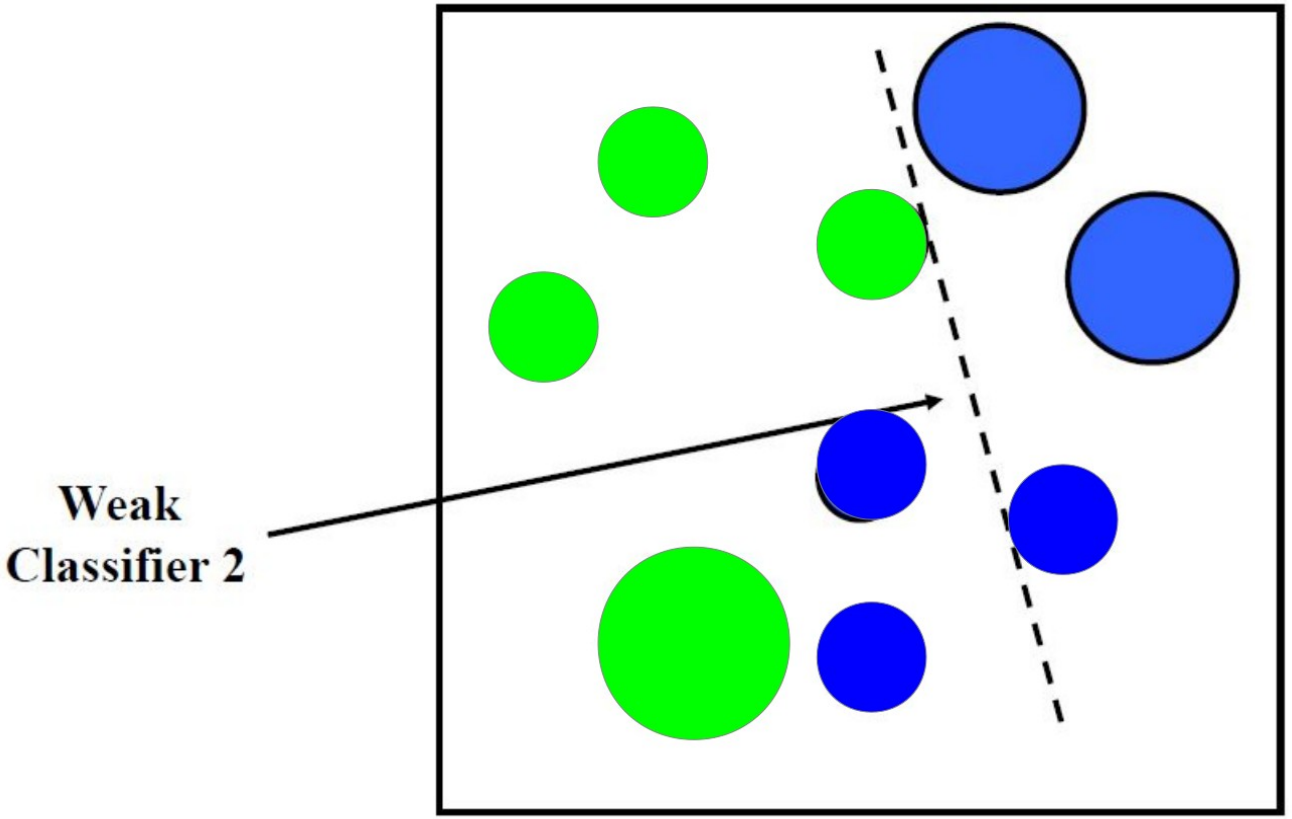


Idea of Boosting:

Select next weak classifier that performs best on weighted training set.

Add more weak classifier until requested training-accuracy is reached.

$$C(x) = \Theta\left(\sum_t h_t(x) + b\right)$$



AdaBoost Algorithm

1. Input the positive and negative training examples along with their labels $\{(\boldsymbol{x}_i, y_i)\}$, where $y_i = 1$ for positive (face) examples and $y_i = -1$ for negative examples.
2. Initialize all the weights to $w_{i,1} \leftarrow \frac{1}{N}$, where N is the number of training examples. (Viola and Jones (2004) use a separate N_1 and N_2 for positive and negative examples.)
3. For each training stage $j = 1 \ldots M$:
   (a) Renormalize the weights so that they sum up to 1 (divide them by their sum).
   (b) Select the best classifier $h_j(\boldsymbol{x}; f_j, \theta_j, s_j)$ by finding the one that minimizes the weighted classification error

$$e_j = \sum_{i=0}^{N-1} w_{i,j} e_{i,j}, \tag{14.3}$$

$$e_{i,j} = 1 - \delta(y_i, h_j(\boldsymbol{x}_i; f_j, \theta_j, s_j)). \tag{14.4}$$

   For any given f_j function, the optimal values of (θ_j, s_j) can be found in linear time using a variant of weighted median computation (Exercise 14.2).
   (c) Compute the modified error rate β_j and classifier weight α_j,

$$\beta_j = \frac{e_j}{1 - e_j} \quad \text{and} \quad \alpha_j = -\log \beta_j. \tag{14.5}$$

   (d) Update the weights according to the classification errors $e_{i,j}$

$$w_{i,j+1} \leftarrow w_{i,j} \beta_j^{1-e_{i,j}}, \tag{14.6}$$

   i.e., downweight the training samples that were correctly classified in proportion to the overall classification error.
4. Set the final classifier to

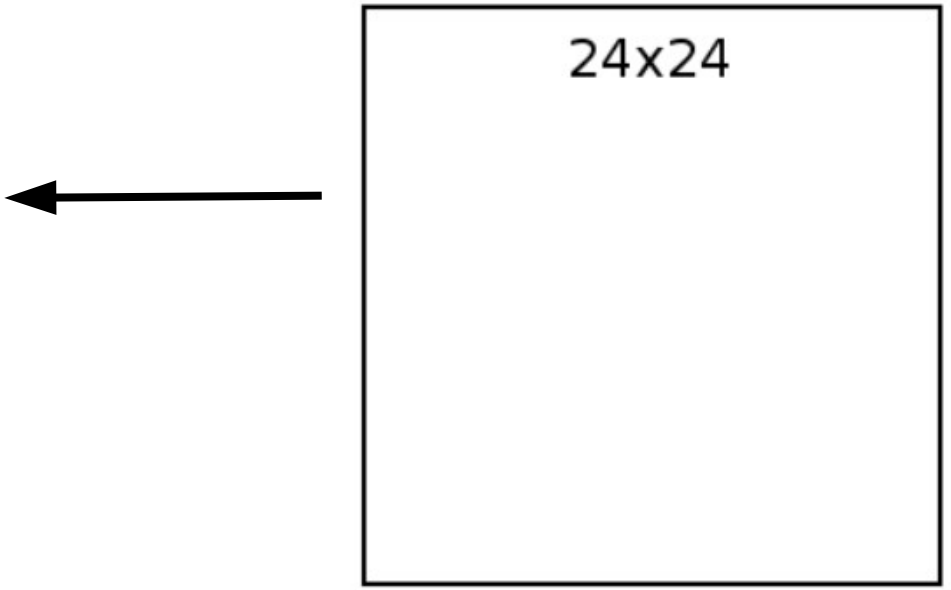
$$h(\boldsymbol{x}) = \operatorname{sign}\left[\sum_{j=0}^{m-1} \alpha_j h_j(\boldsymbol{x})\right]. \tag{14.7}$$

Algorithm 14.1 The AdaBoost training algorithm, adapted from Hastie, Tibshirani, and Friedman (2001), Viola and Jones (2004), and Bishop (2006).

Observations: Face-Images

- Faces are rare! (0-10 faces per image)
- 1000 times as many non-faces
- 10000-50000 locations/scales

We need a simple and quick way to classify and withdraw “non-faces”

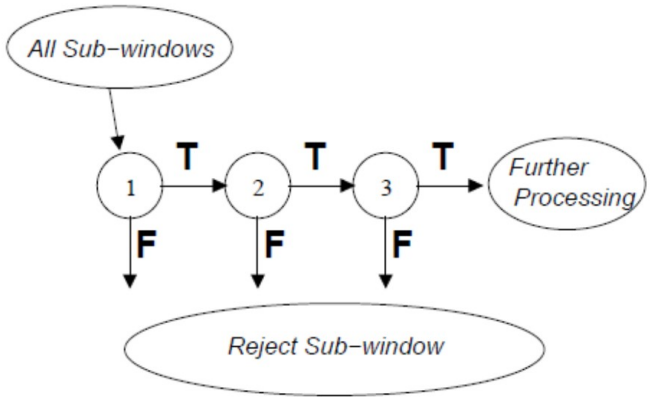


Build a cascade of classifiers, each stage involving more and different features.

(Idea: Faces – which are rare – can take much more processing time.)

Cascade of Classifiers

Keep a high speed: Exploit a cascade (layers) of subsequent classifiers



Each layer of the cascade consists of a classifier with increasing complexity (here a higher number of involved features).

Subsequent classifiers are trained using the training examples which pass through all the previous classifiers.

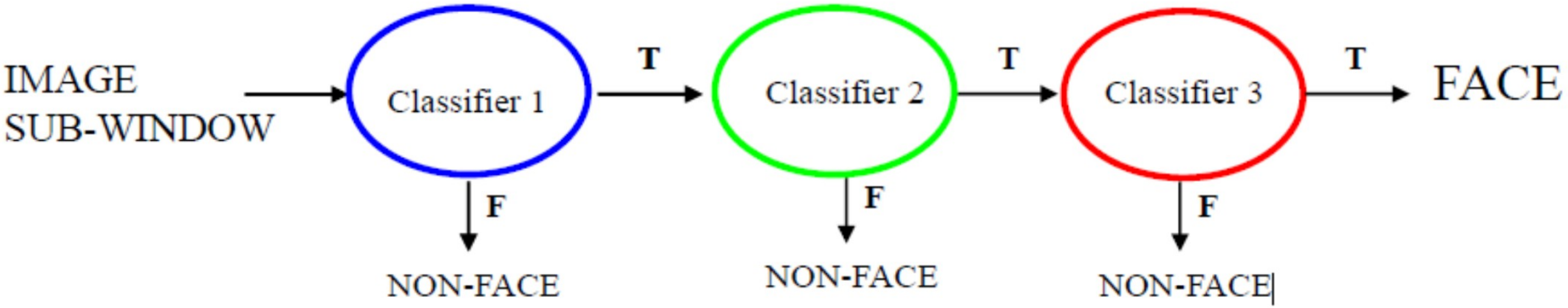
The cascade rejects many of negative sub-windows (non-faces) at high levels

→ no need for further processing

Many omitted details: image normalization, detector window shifting, detector window scaling, details on the used classifiers

Cascading Classifiers

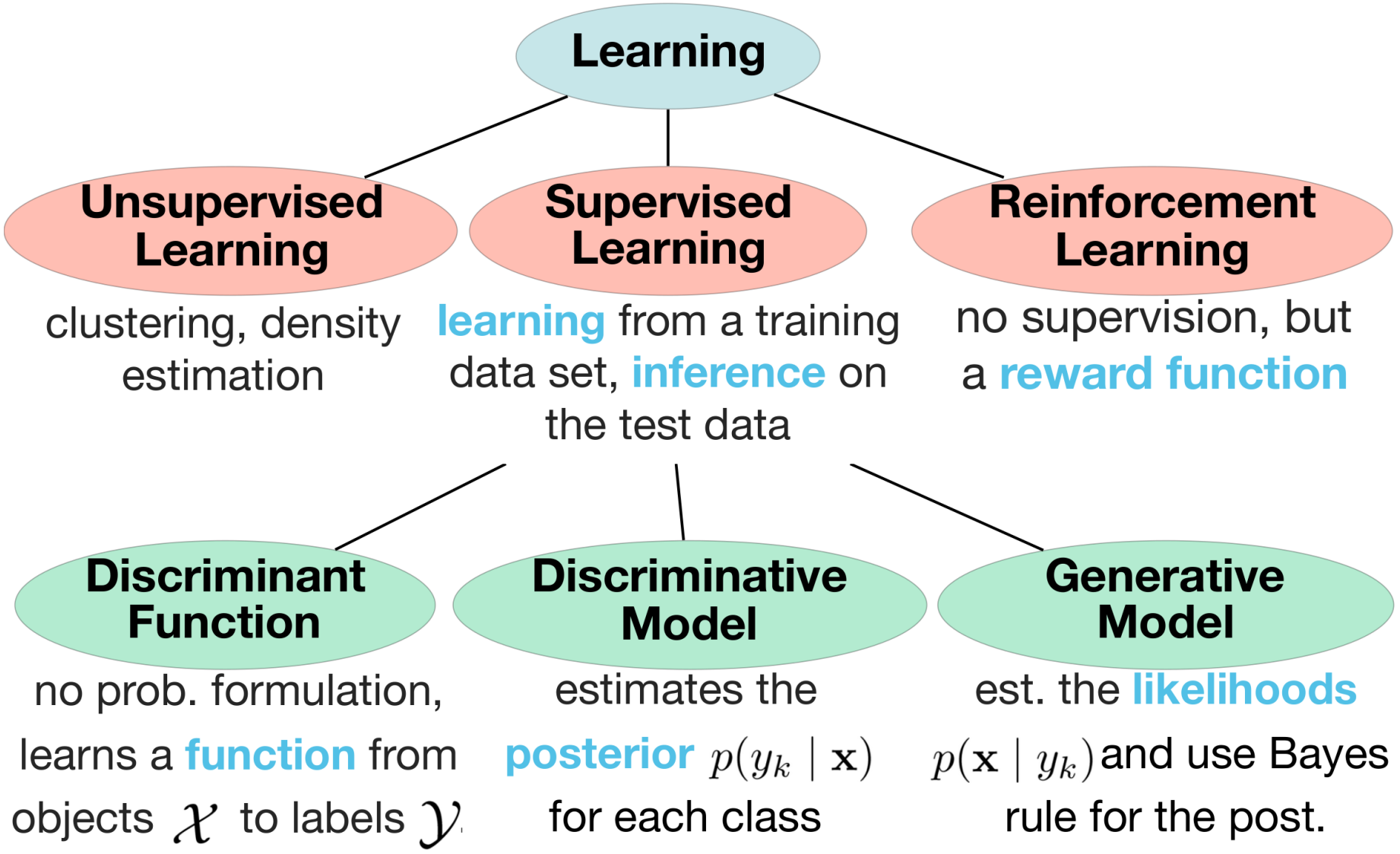
- We start with simple classifiers which reject many of the negative sub-windows while detecting almost all positive sub-windows
- Positive results from the first classifier triggers the evaluation of a second (more complex) classifier, and so on
- A negative outcome at any point leads to the immediate rejection of the sub-window



Strategy at lower stages: remove non-faces

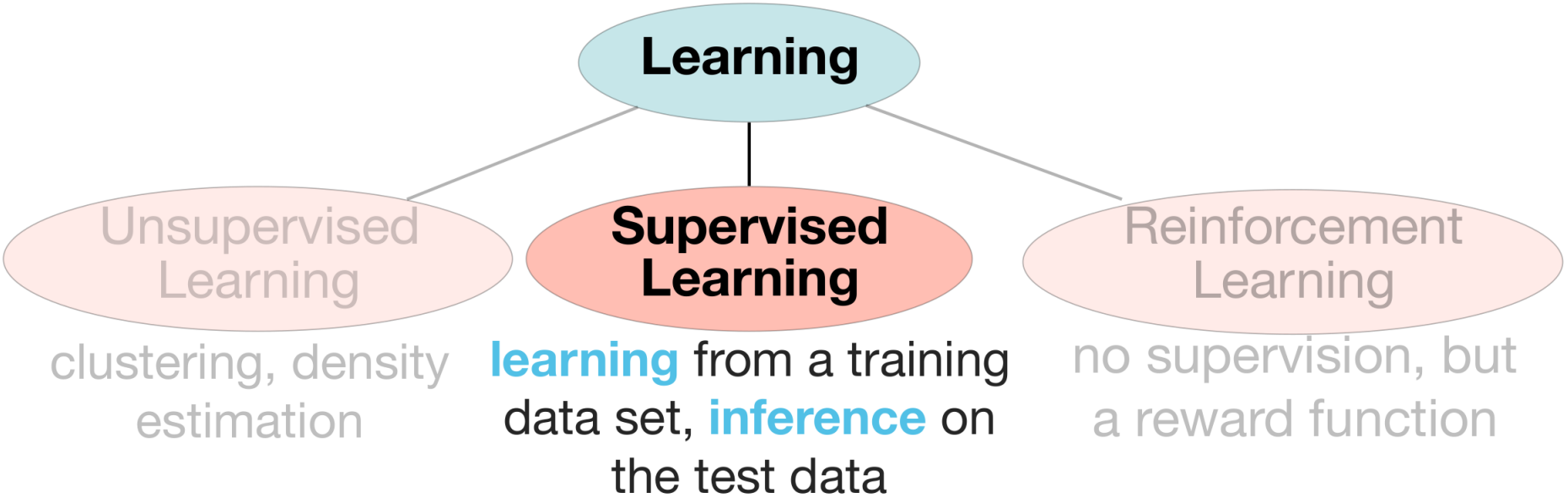
Strategy higher stages: use more efforts to find faces

Categories of Learning



R. Triebel (TUM)

Supervised Learning

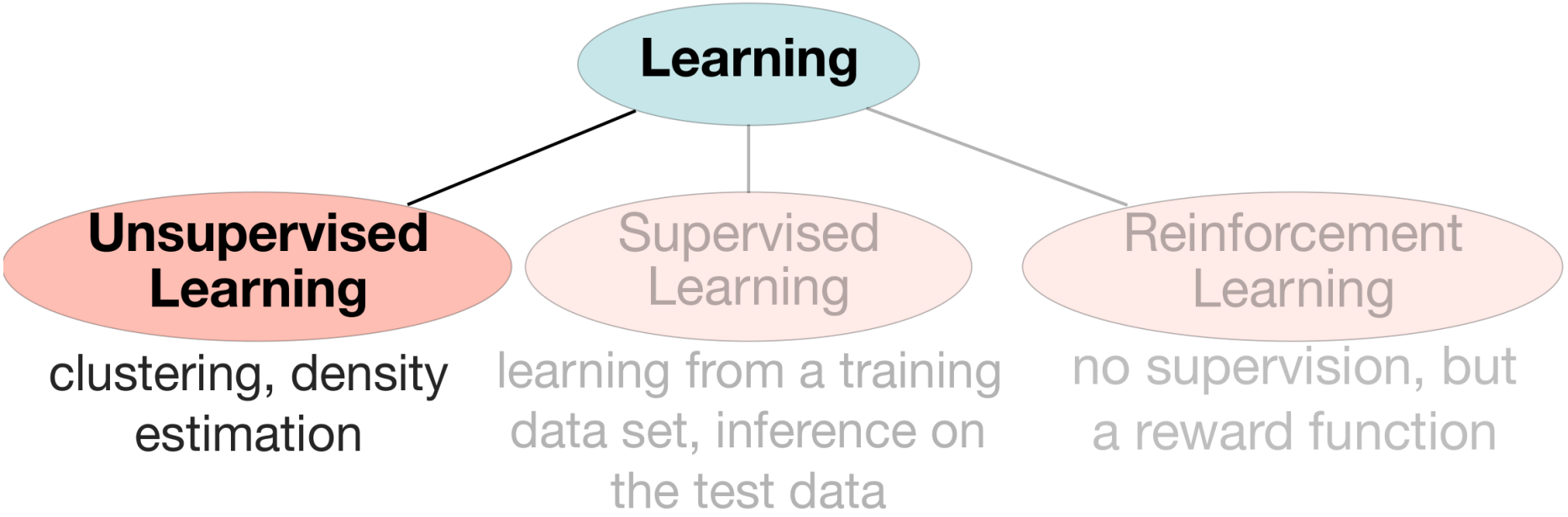


Supervised learning methods used in computer vision include:

- Regression
- Conditional Random Fields
- Boosting
- Machine Learning for Computer Vision
- Support Vector Machines
- Gaussian Processes
- Hidden Markov Models

R. Triebel (TUM)

Unsupervised Learning

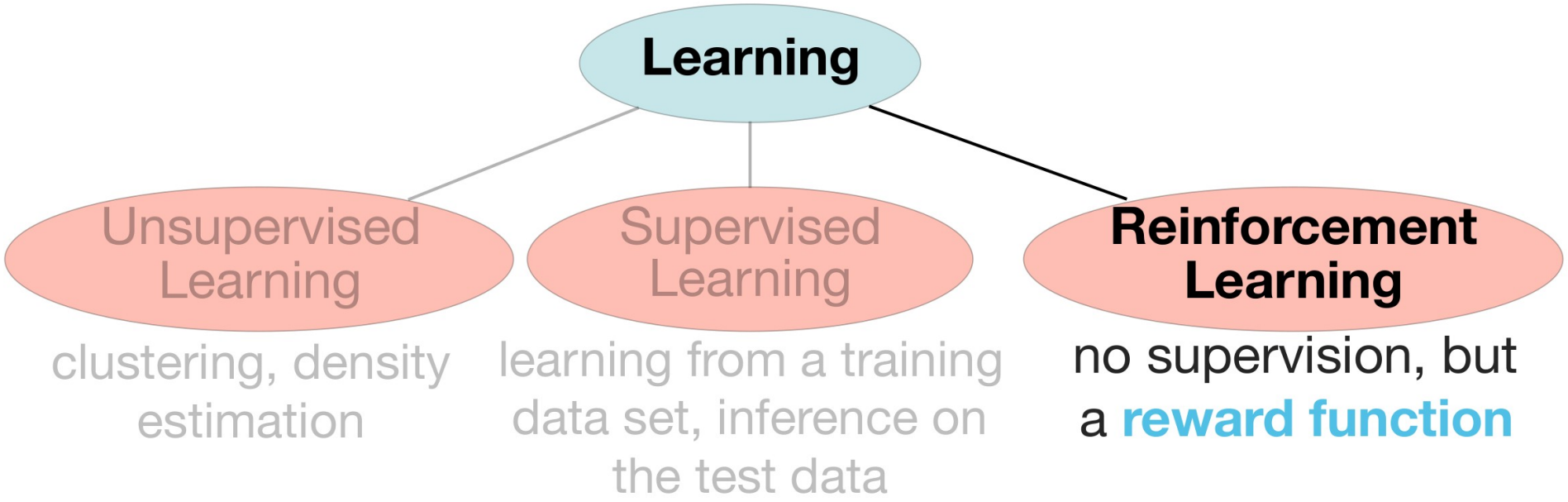


In unsupervised learning, there is no ground truth information given.

Most Unsupervised Learning methods are based on **Clustering.**

R. Triebel (TUM)

Reinforcement Learning

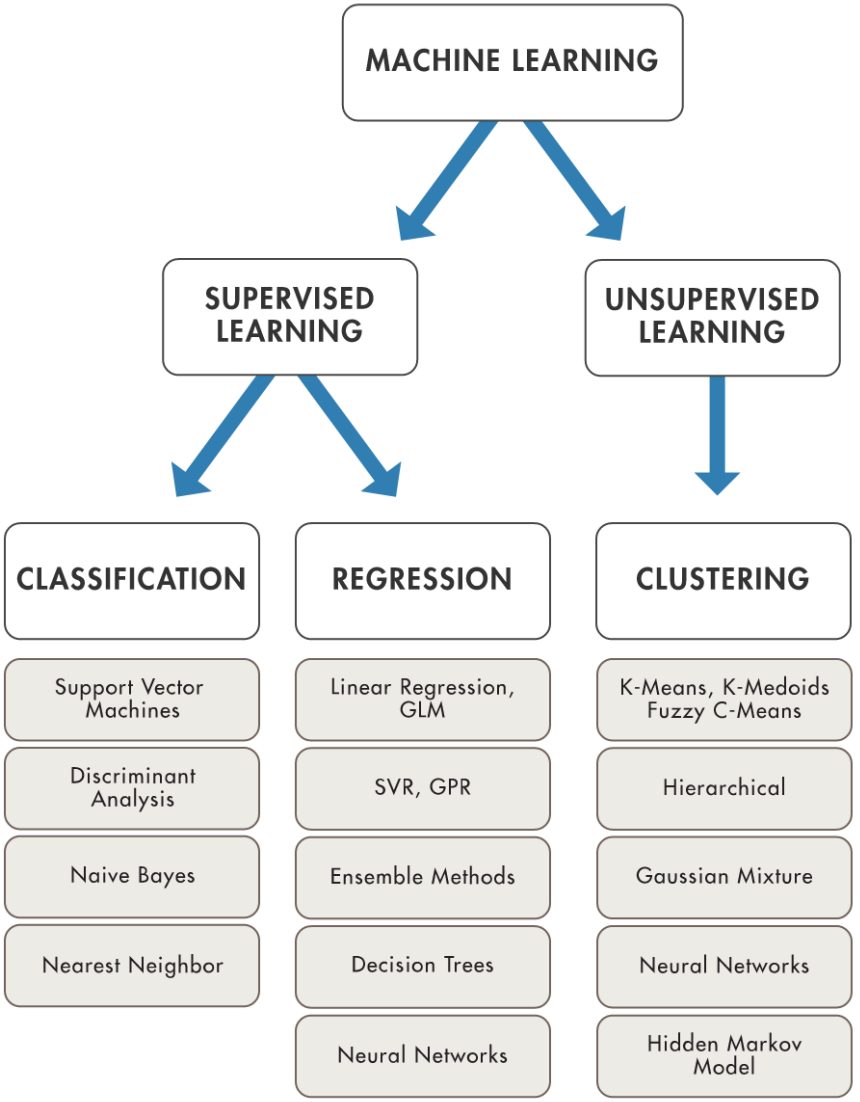


Reinforcement Learning requires an action

- the reward defines the quality of an action
- mostly used in robotics (e.g. manipulation)
- can be dangerous, actions need to be “tried out”

R. Triebel (TUM)

Methods - Categories



Recognition Techniques

- Classifier:
  - Support Vector Machines (SVM)
- Cluster-Analysis:
  - k-means, mean-shift

Clustering

- Clustering is a common “recognition” task in Computer Vision.
- Examples include:
  - Classification
  - Segmentation
  - Vector quantization
- A classic and simple approach from statistical data analysis is: **k-means**

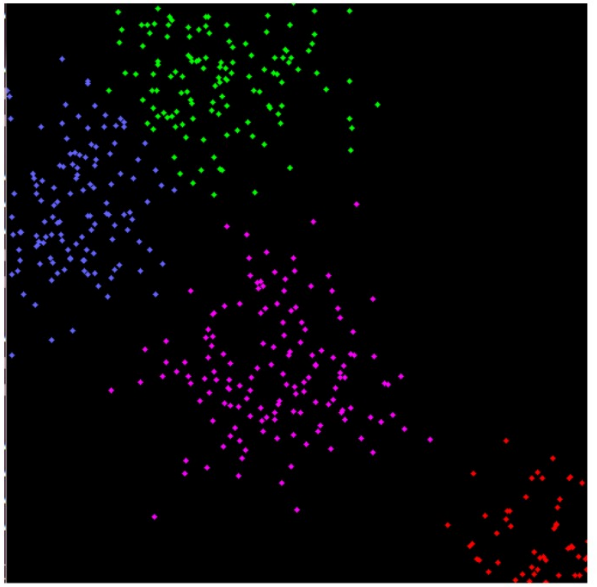
Recall: What do we call Features?

- Nearly every image information which can help us to solve a CV related task.
- Feature detection refers to the abstraction of image information
- Examples:
  - Color, shape or texture information from images.
  - Salient points (Corner detectors)
  - More advanced description for points of interest (SIFT)
  - A small image patch (interpret pixel-array as vector)

Recall: Features

- Feature description
  - Shape description
    - Boundary or Region based
    - Moments
    - Fourier-descriptor
  - Feature detection
    - Harris-Corner detector
    - SIFT-Features (Variants: SURF, BRIEF, ORB, etc.)
- Idea: Represent the obtained information as a (high-dimensional) vector.

→ Point in the feature space

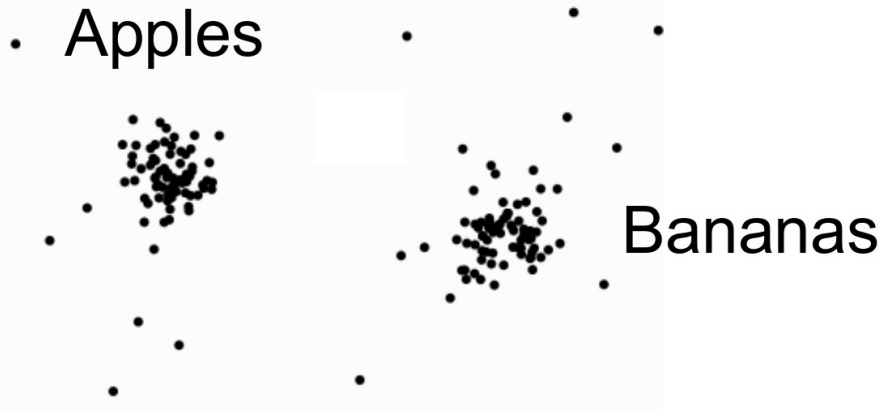


Features

We need numerical expressions
to stack n measurements in a n-dimensional vector.

Here n=2:

$$x \in \mathbb{R}^n$$



“The hope is that objects with the same properties cluster.”

K-Means Clustering

Select k (random) data points as initial “cluster centers”

Until the cluster centers are unchanged

Allocate each data point to nearest cluster center

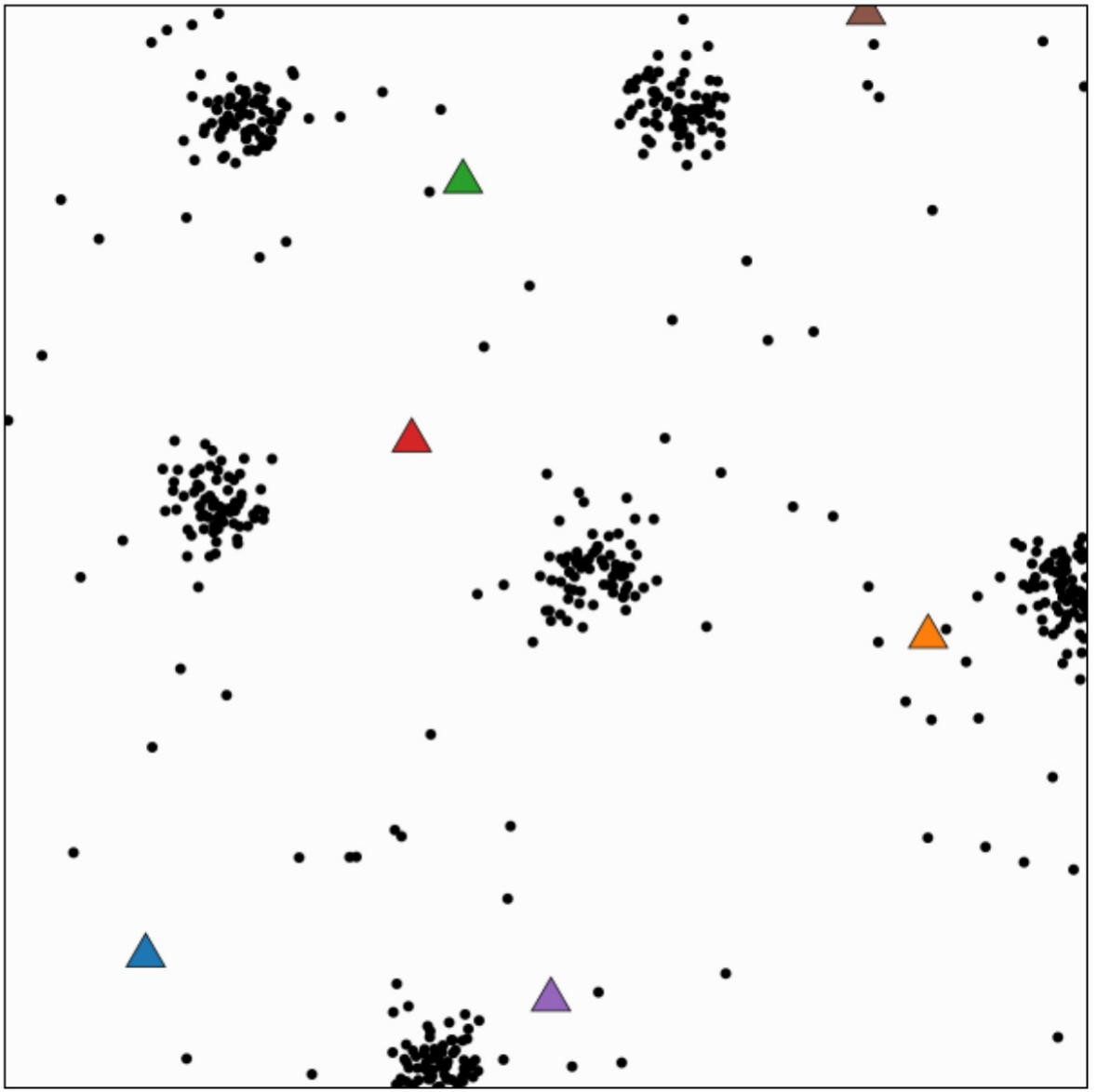
Now ensure that every cluster has at least one data point; possible techniques for doing this include f.e. supplying empty clusters with a point chosen at random from points far from their cluster center.

Replace the cluster centers with the mean of the elements in their clusters.

end

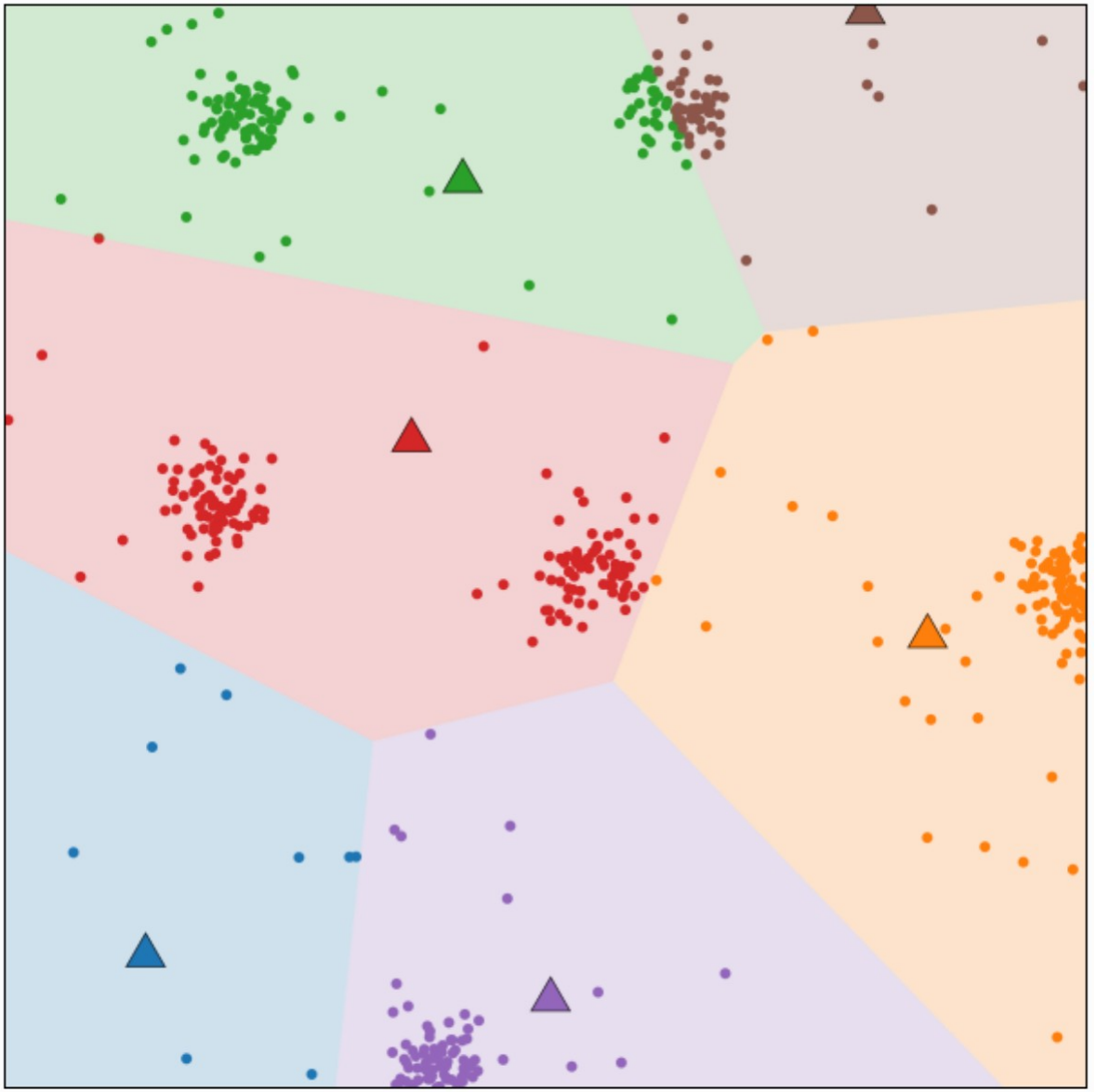
K-Means

- Set number of clusters (k=6)
- Choose k random centers
- Assign all data points to nearest center
- Compute new data centers
- Repeat ...



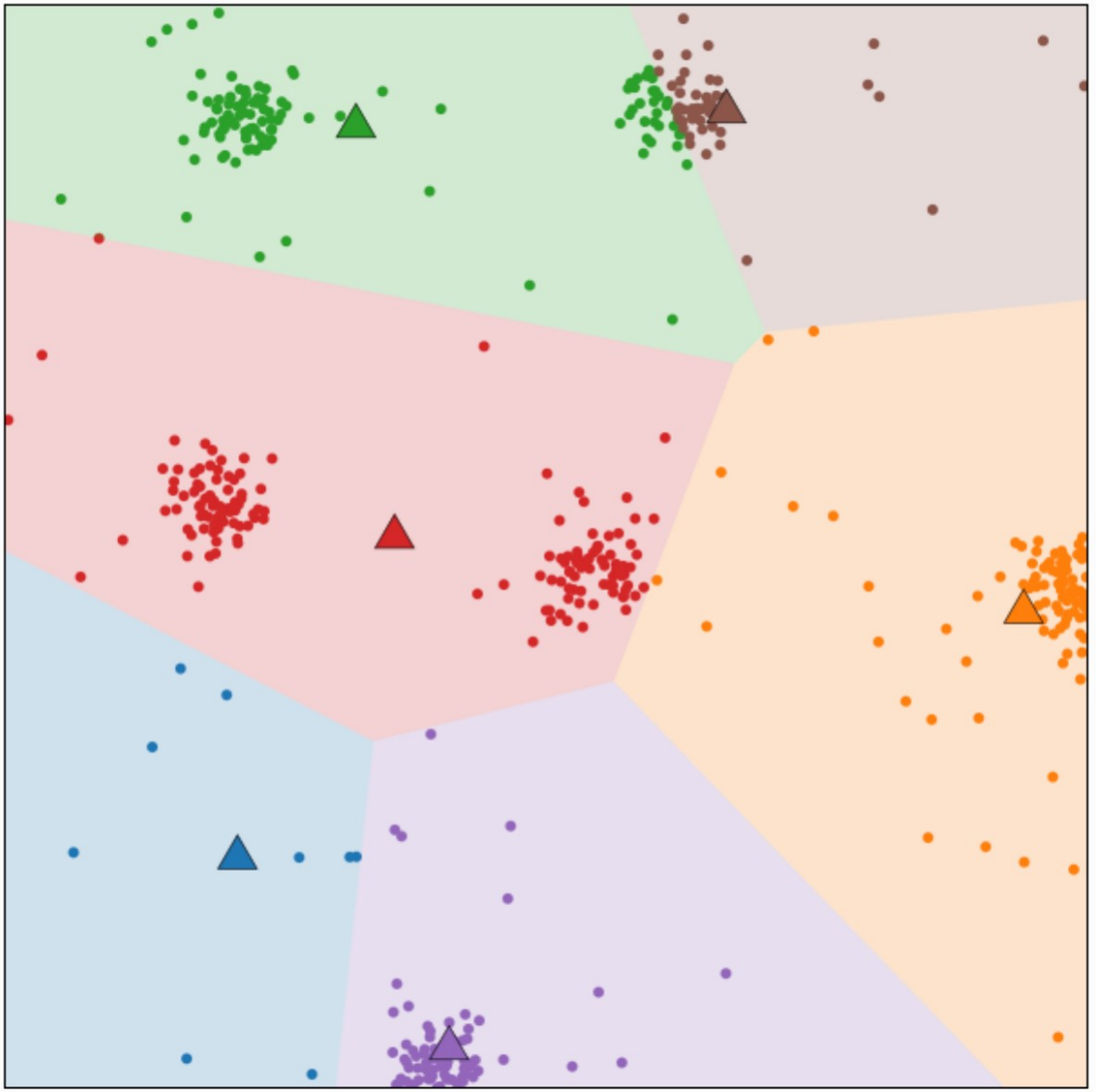
K-Means

- Set number of clusters (k=6)
- Choose k random centers
- Assign all data points to nearest center
- Compute new data centers
- Repeat ...



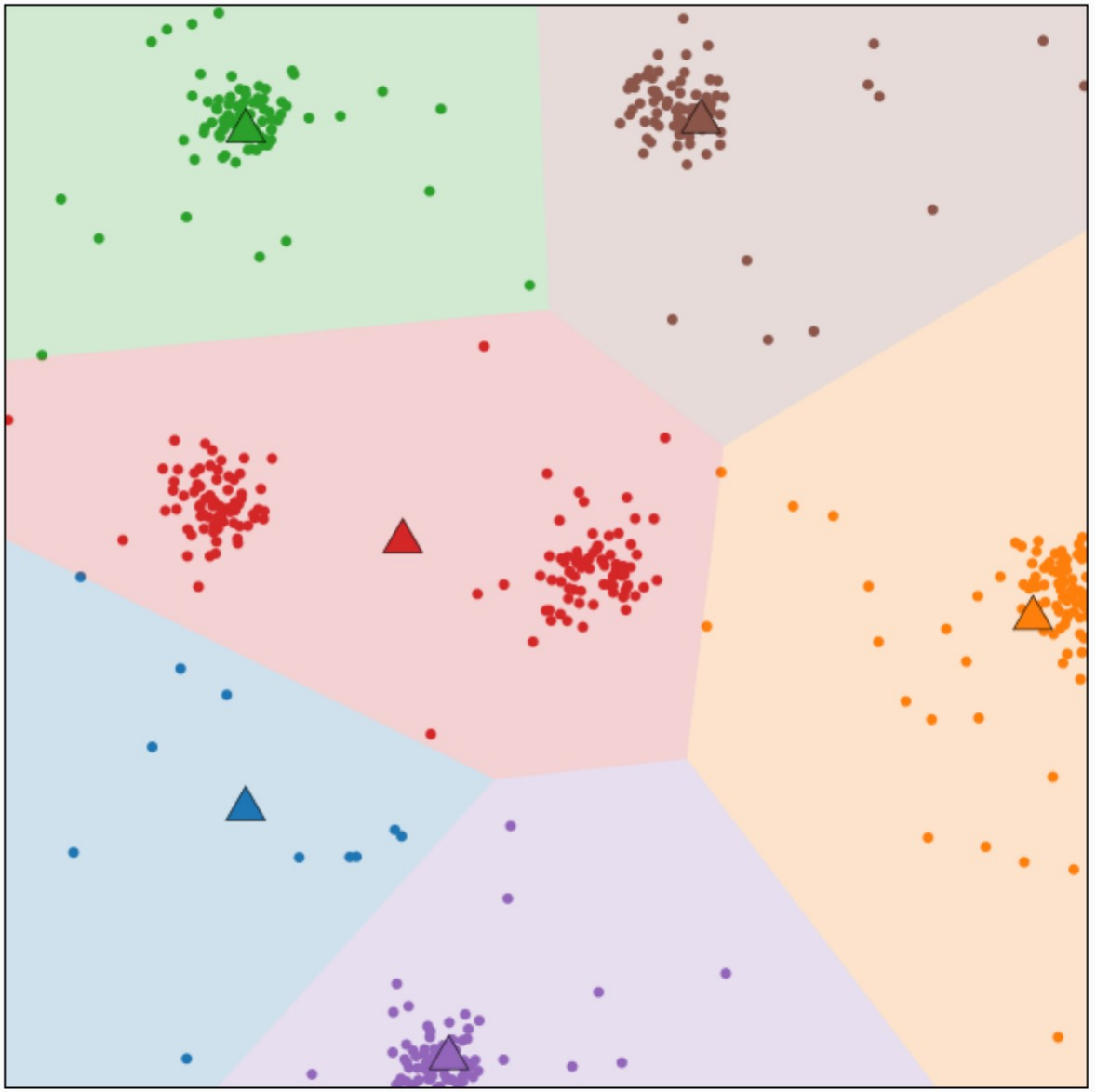
K-Means

- Set number of clusters (k=6)
- Choose k random centers
- Assign all data points to nearest center
- Compute new data centers
- Repeat ...



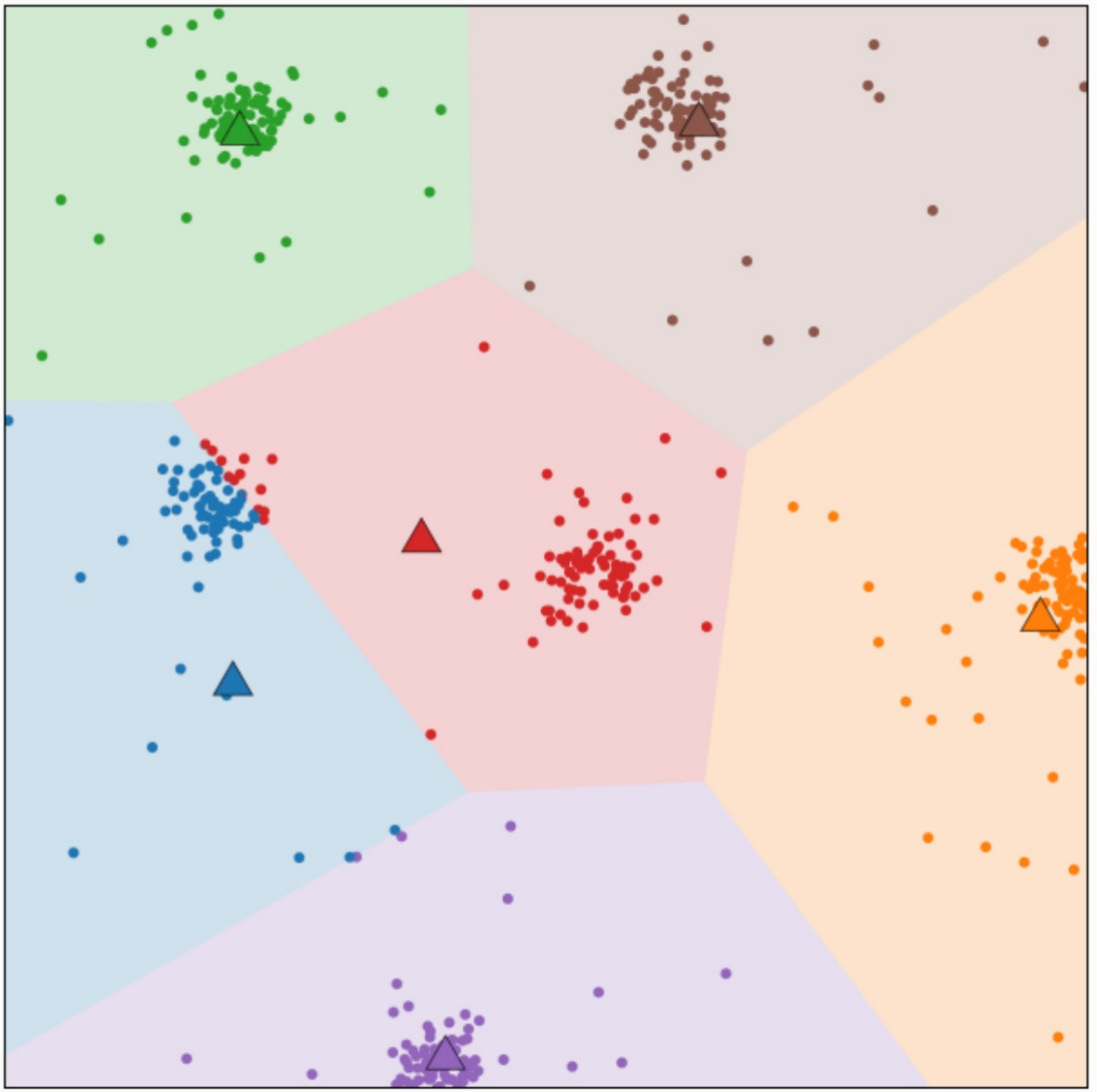
K-Means

- Set number of clusters (k=6)
- Choose k random centers
- Assign all data points to nearest center
- Compute new data centers
- Repeat ...



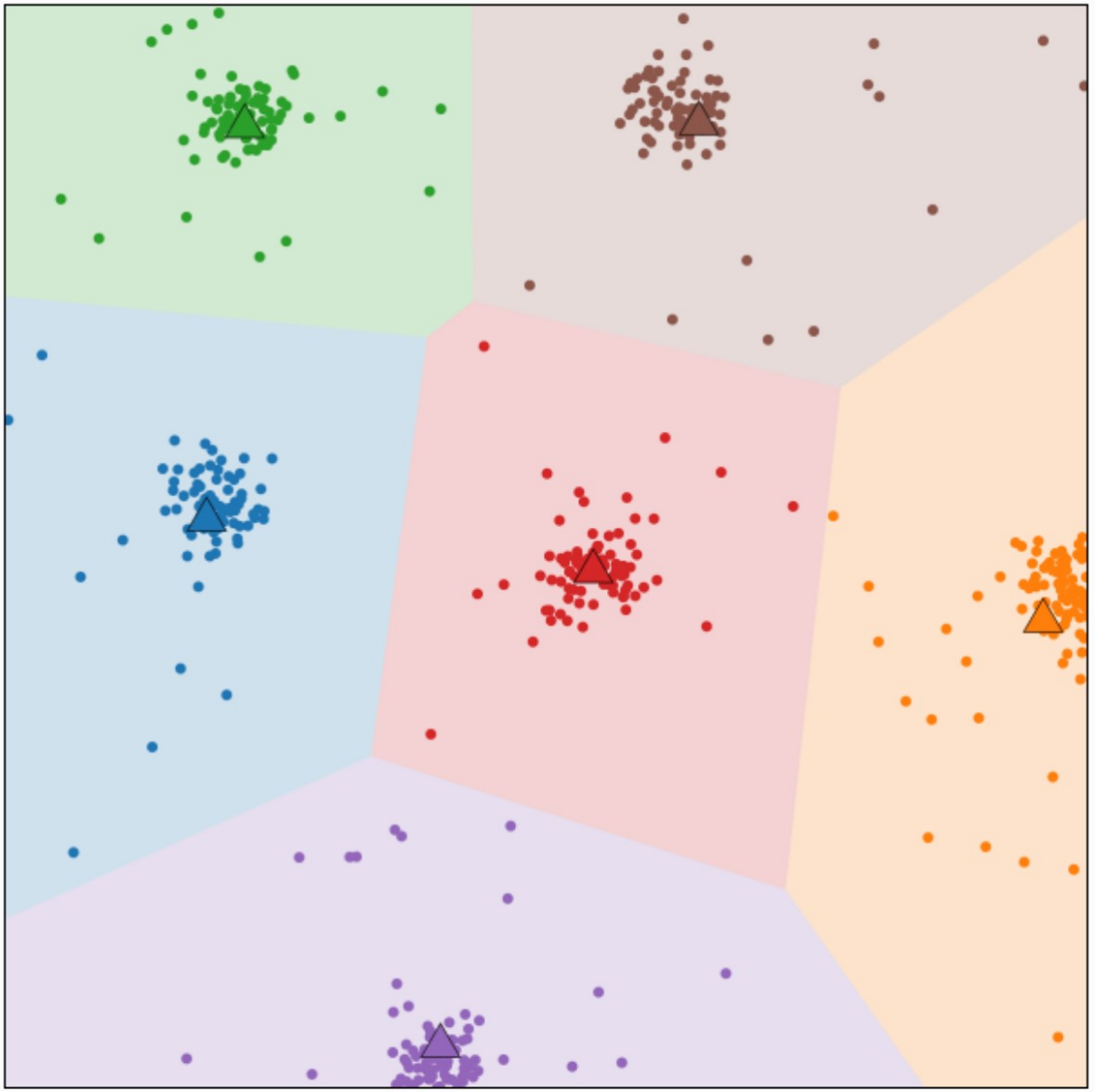
K-Means

- Set number of clusters (k=6)
- Choose k random centers
- Assign all data points to nearest center
- Compute new data centers
- Repeat ...



K-Means

- Set number of clusters (k=6)
- Choose k random centers
- Assign all data points to nearest center
- Compute new data centers
- Repeat ...



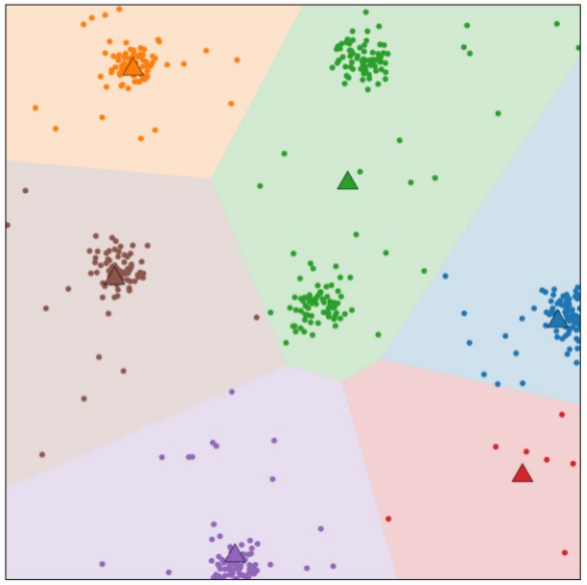
K-Means

+ **Unsupervised** Clustering Method

+ **Simple**, works well for clear convex clusters

- parameter k has to be selected

- can get stuck in a bad local minimum



“Repeat with different start-configurations. Measure the total distance of the data to nearest center.”

K-Means

- Only local optimum is found (often good)
- Lloyds algorithm “linear”

  → O(nkdi): n: data-pts, d:dim, k: clustnum, i:iterations until convergence
- Global Optimum

$$\arg\min_{S} \sum_{i=i}^{k} \sum_{x_j \in S_i} \left\| x_j - \mu_i \right\|^2$$

  difficult to compute
- Variant of so called “expectation-maximization algorithms”

  assignment=expectation

  update of mean=maximization (optimization)

Example K-Means: Color Segmentation

A simple color segmentation using K-Means

K=3

Color vector:

$$c = \begin{pmatrix} r \\ g \\ b \end{pmatrix}$$

→ Simple and effective way, but not really fast ;)

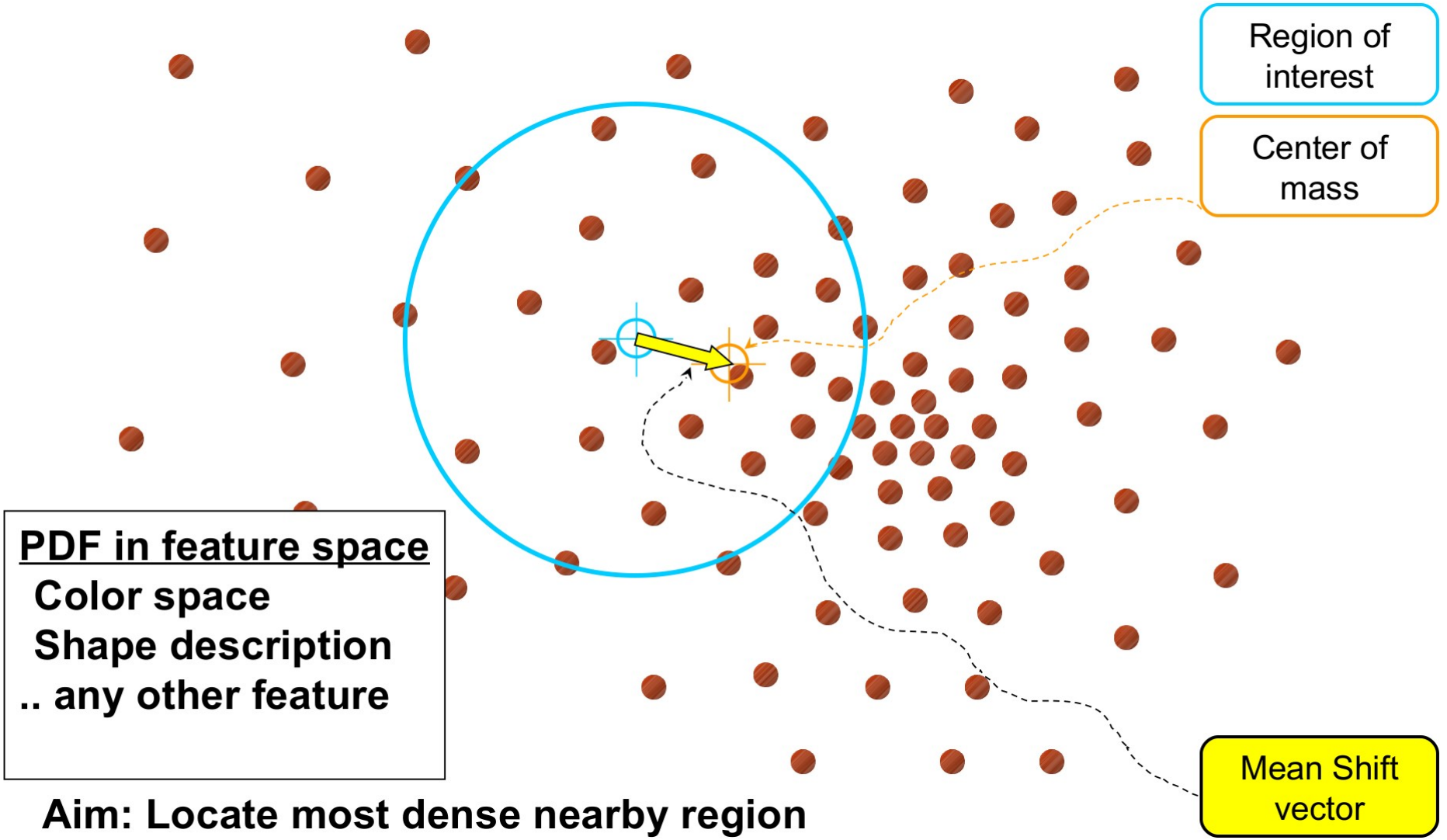
Mean Shift Theory

- What is Mean Shift ?
  - Density Estimation Methods
  - A few details Mean Shift
  - Mean shift properties
- Applications
  - Clustering
  - Discontinuity Preserving Smoothing

Mean shift: A robust approach toward feature space analysis. D. Comaniciu P. Meer, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, No 5. (2002), pp. 603-619.

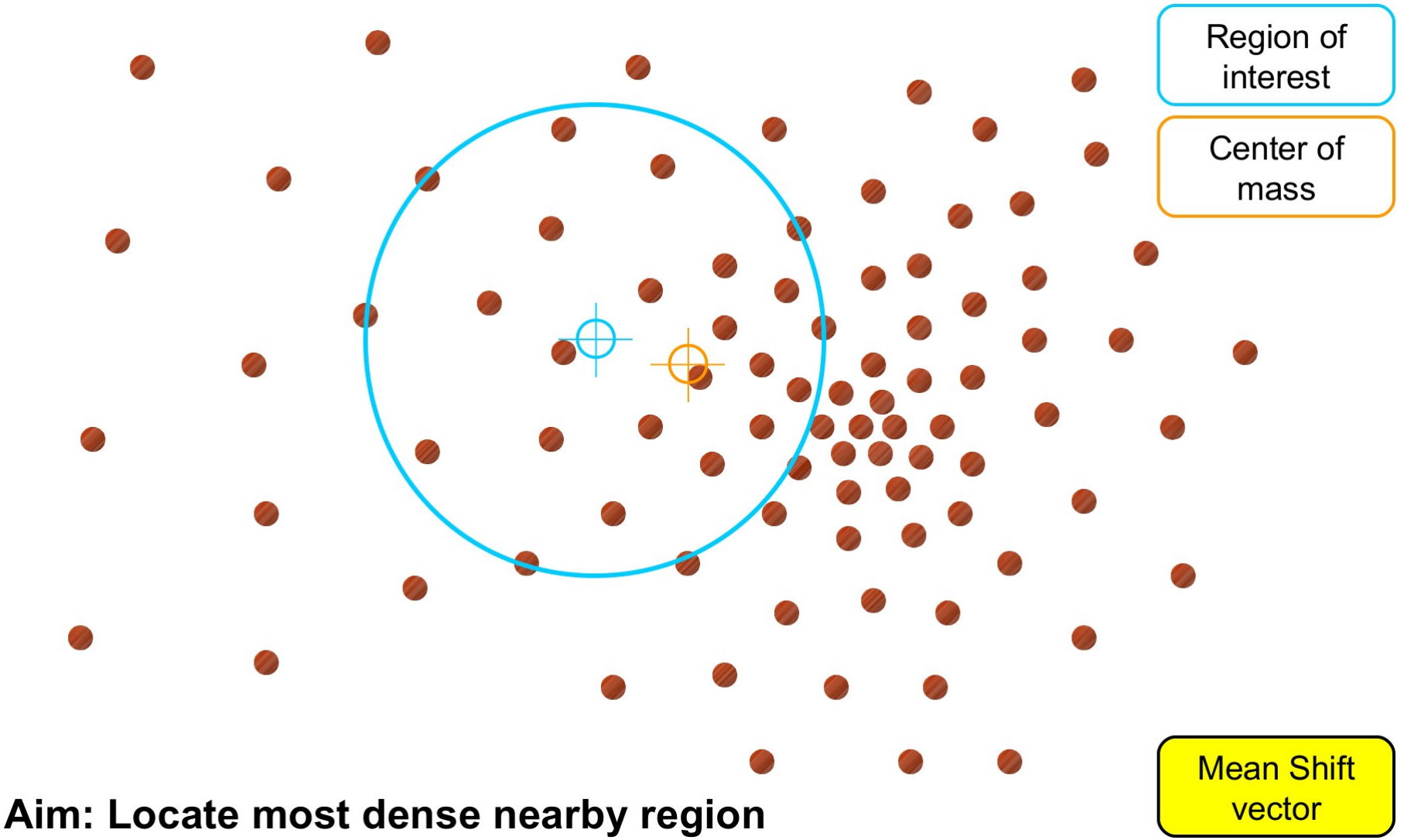
Slide Credit: Yaron Ukrainitz & Bernard Sarel

Basic Idea: Mean Shift



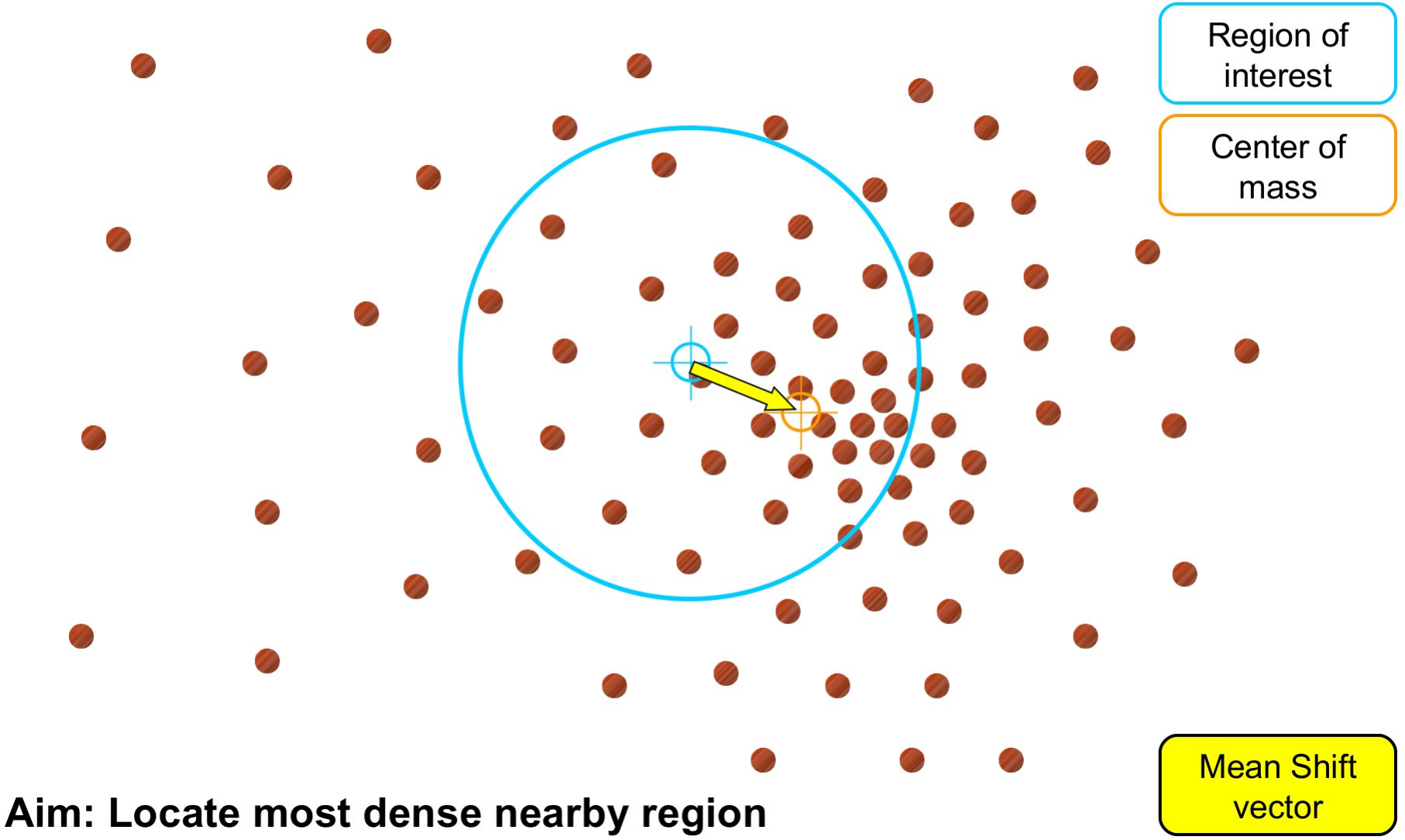
Courtesy: Christian Schellewald, 2020

Basic Idea: Mean Shift



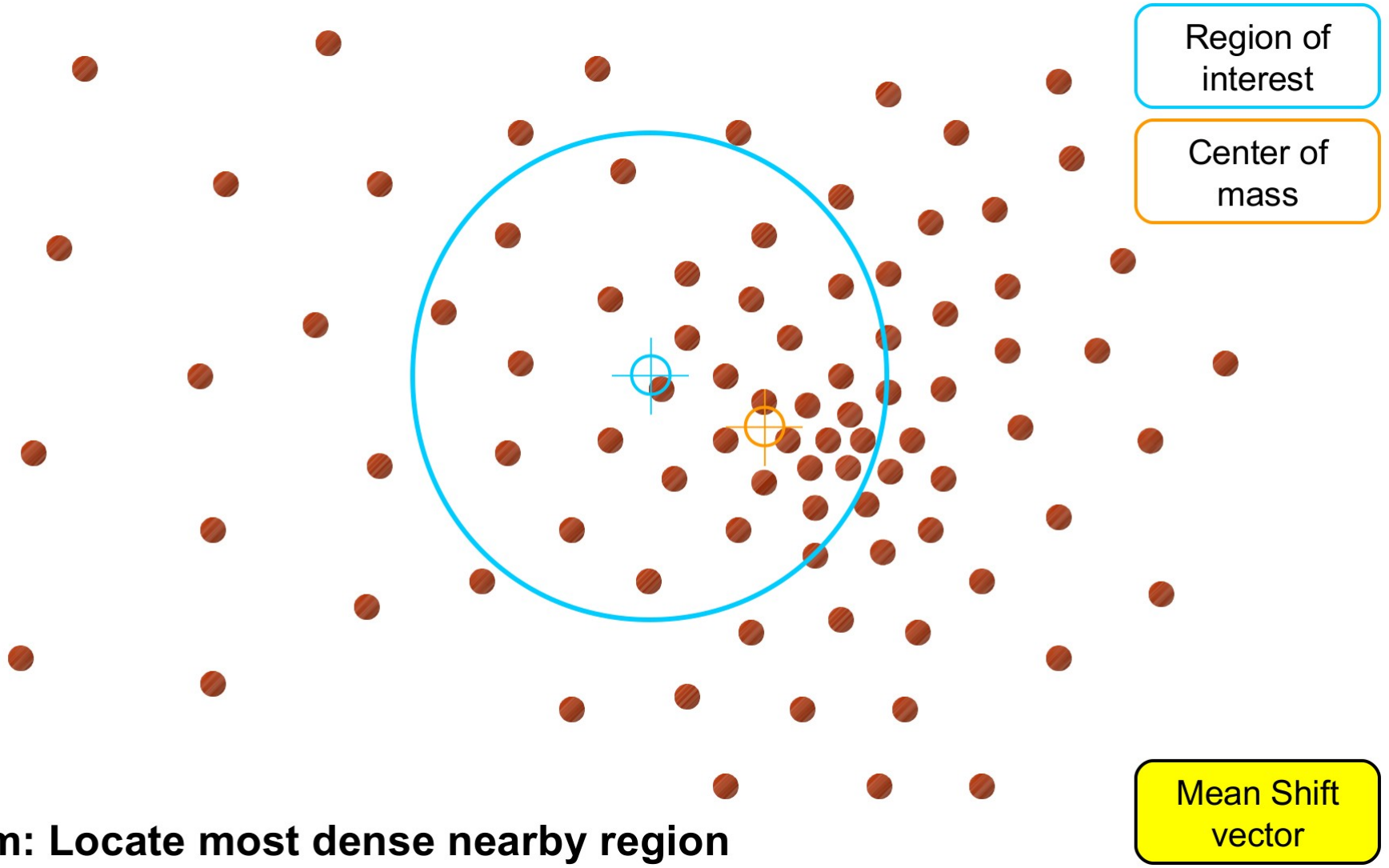
Courtesy: Christian Schellewald, 2020

Basic Idea: Mean Shift



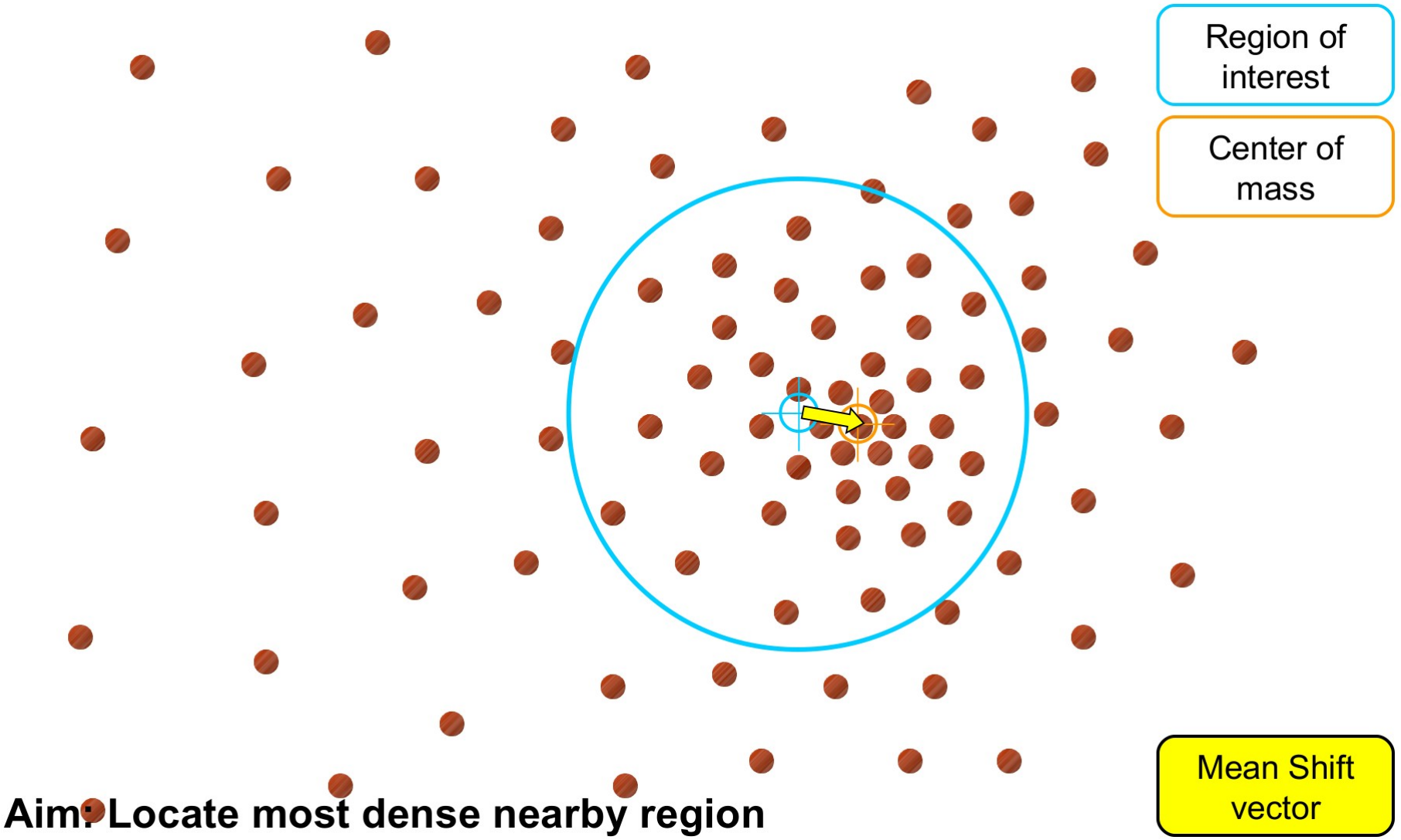
Aim: Locate most dense nearby region

Basic Idea: Mean Shift



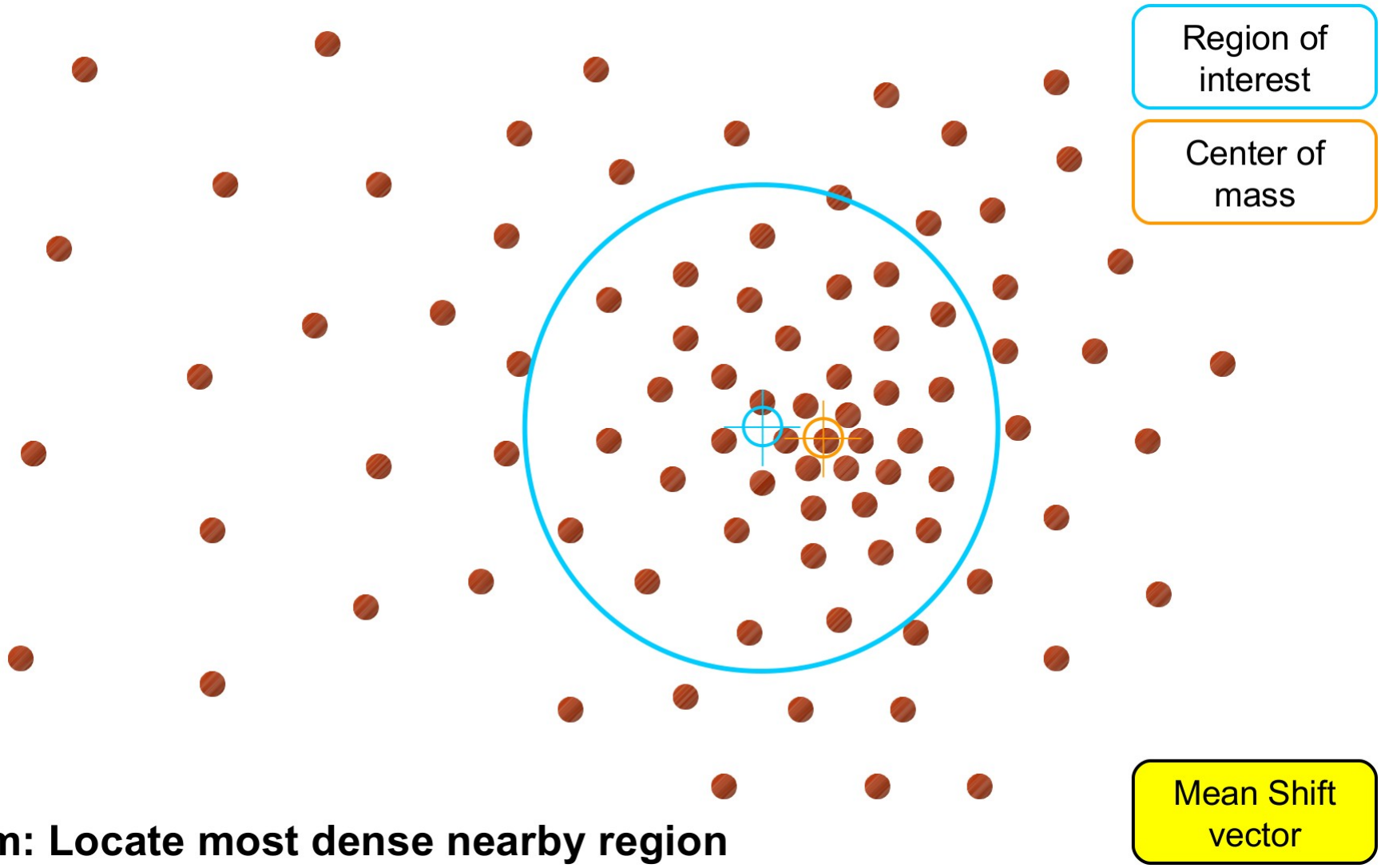
Courtesy: Christian Schellewald, 2020

Basic Idea: Mean Shift



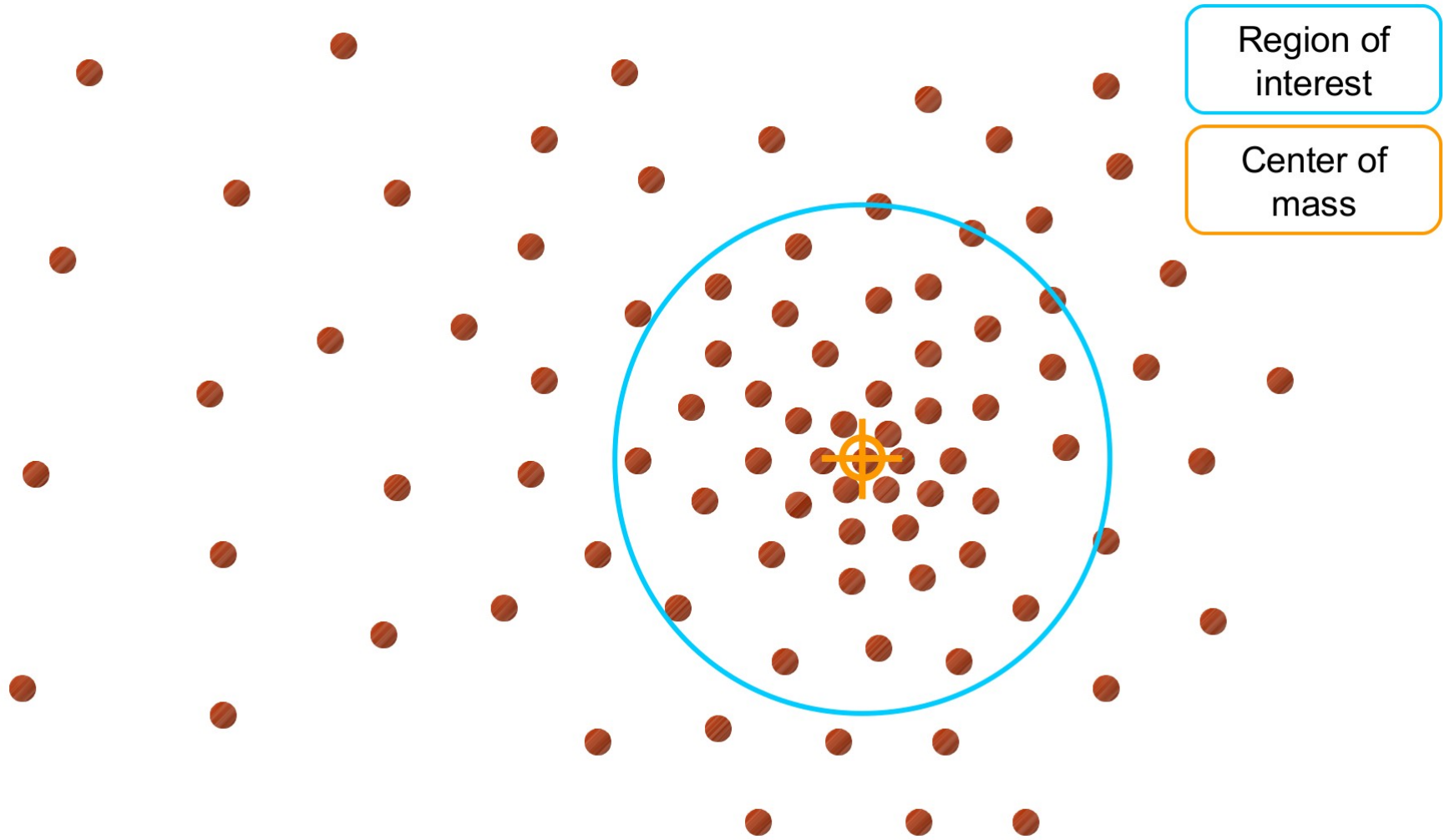
Courtesy: Christian Schellewald, 2020

Basic Idea: Mean Shift



Courtesy: Christian Schellewald, 2020

Basic Idea: Mean Shift

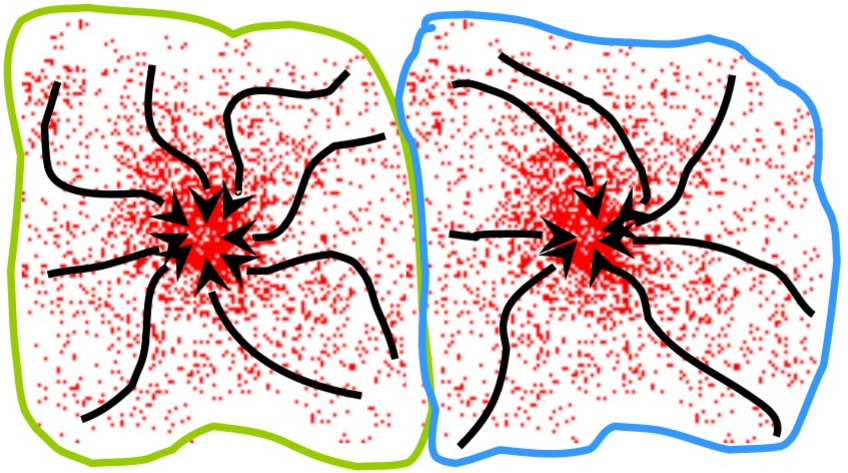


Aim: Locate most dense nearby region

Courtesy: Christian Schellewald, 2020

Clustering

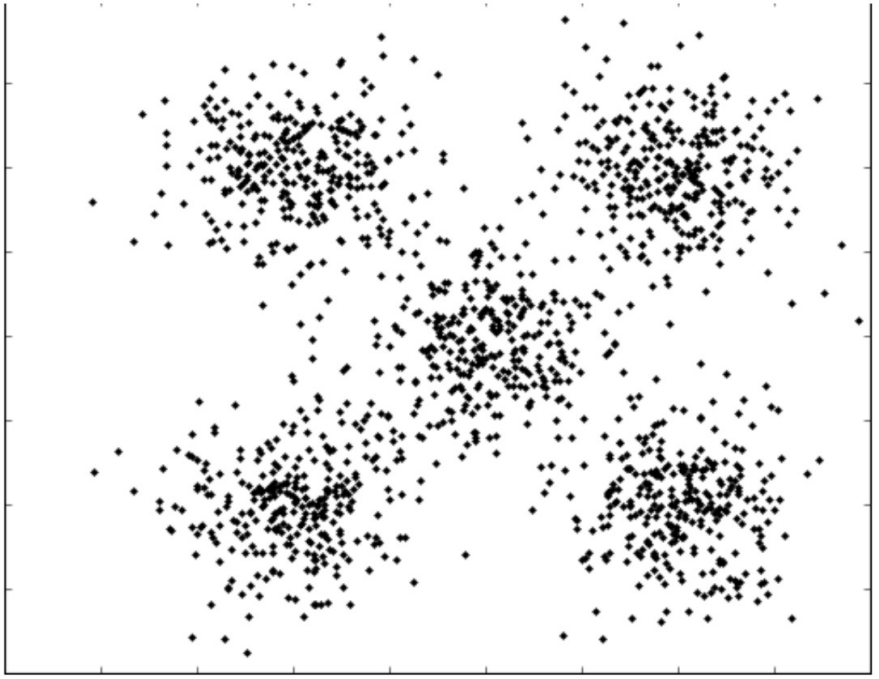
Cluster: All data points in the attraction basin of a mode

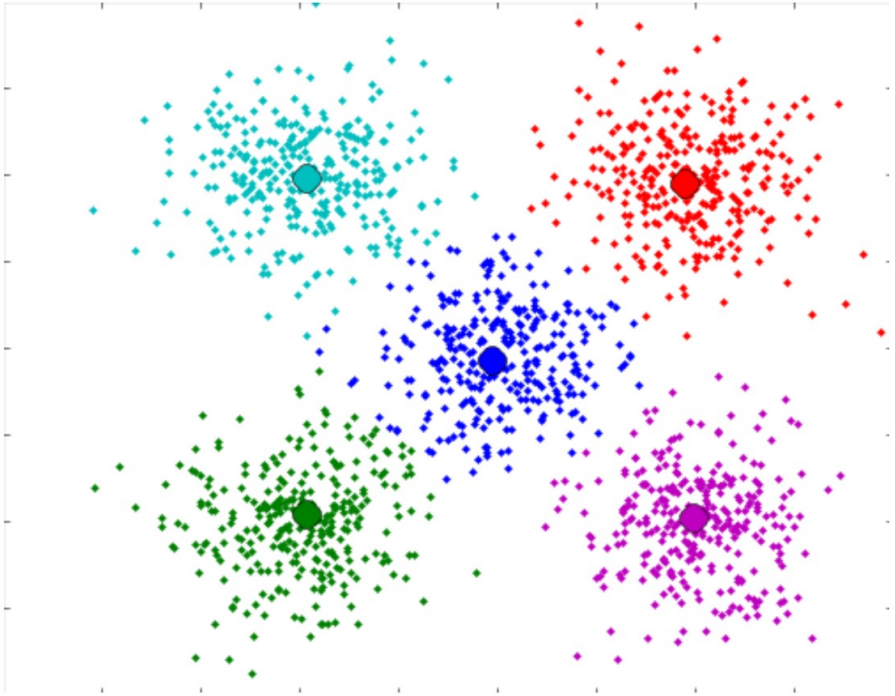


Attraction basin: the region for which all trajectories lead to the same mode

Mean-Shift 2D Example

The number of clusters is automatically determined!
(In K-means this is a parameter)



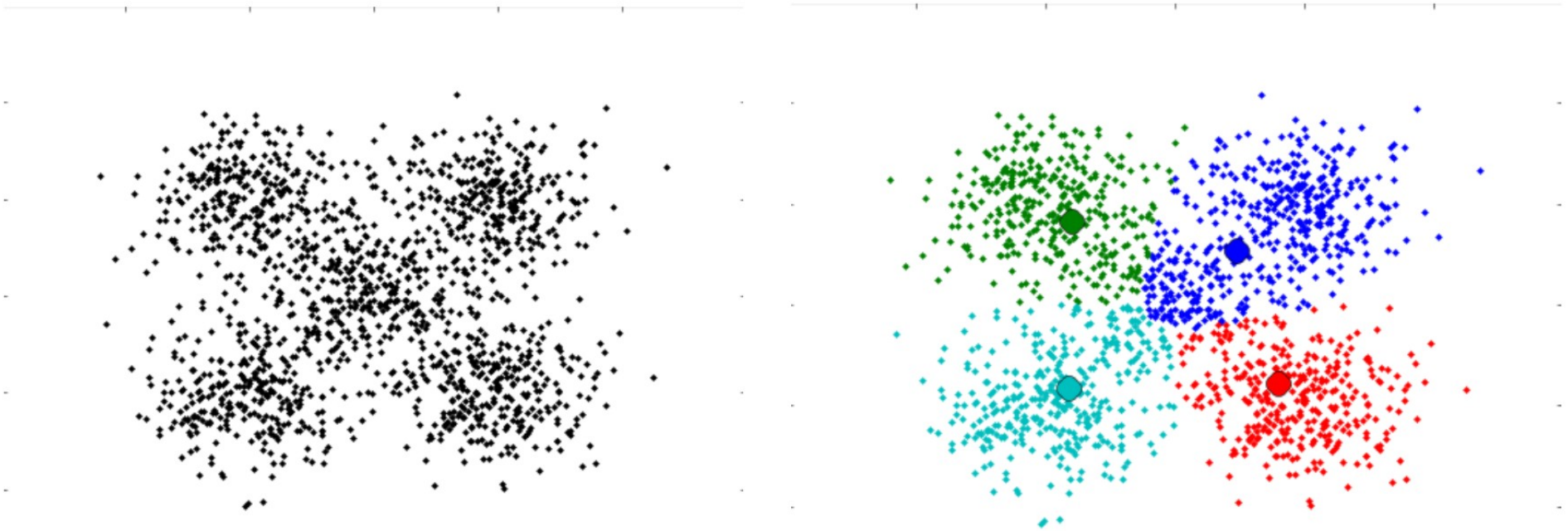


Courtesy: Christian Schellewald, 2020

Mean-Shift 2D Example

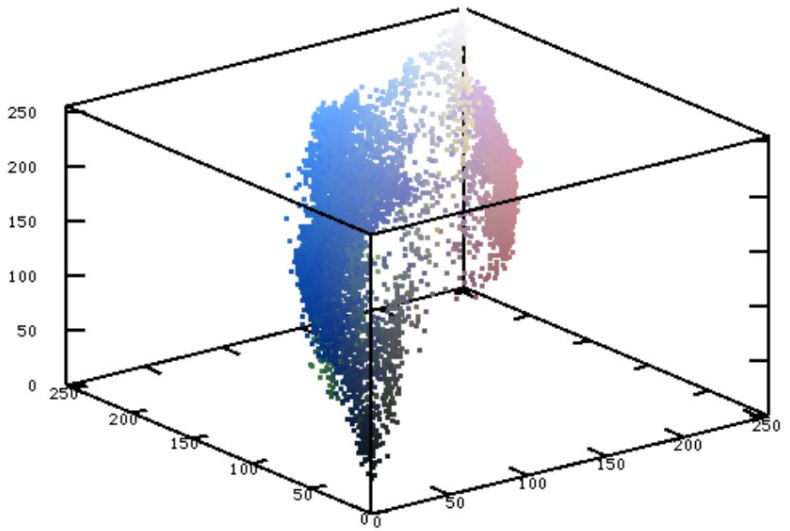
Only 4 clusters detected (big points= estimated max density)

→ Kernel-size

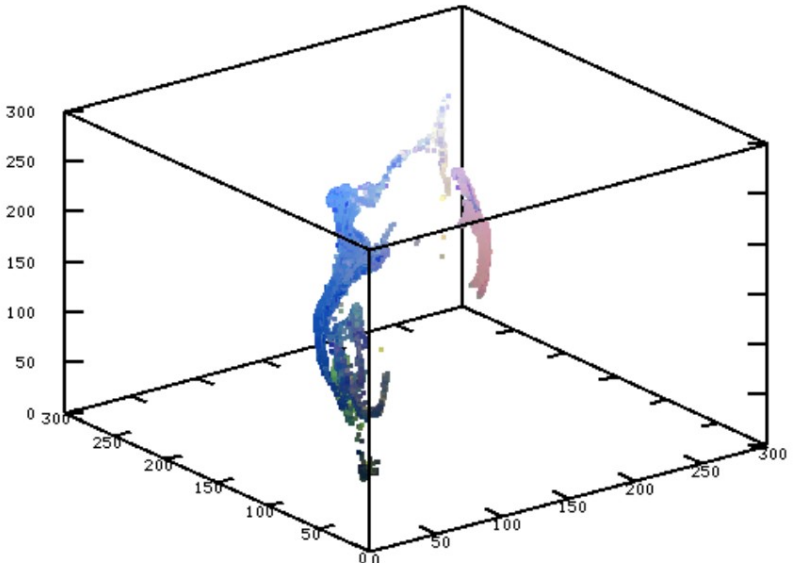


Courtesy: Christian Schellewald, 2020

Mean Shift Example



RGB-color space



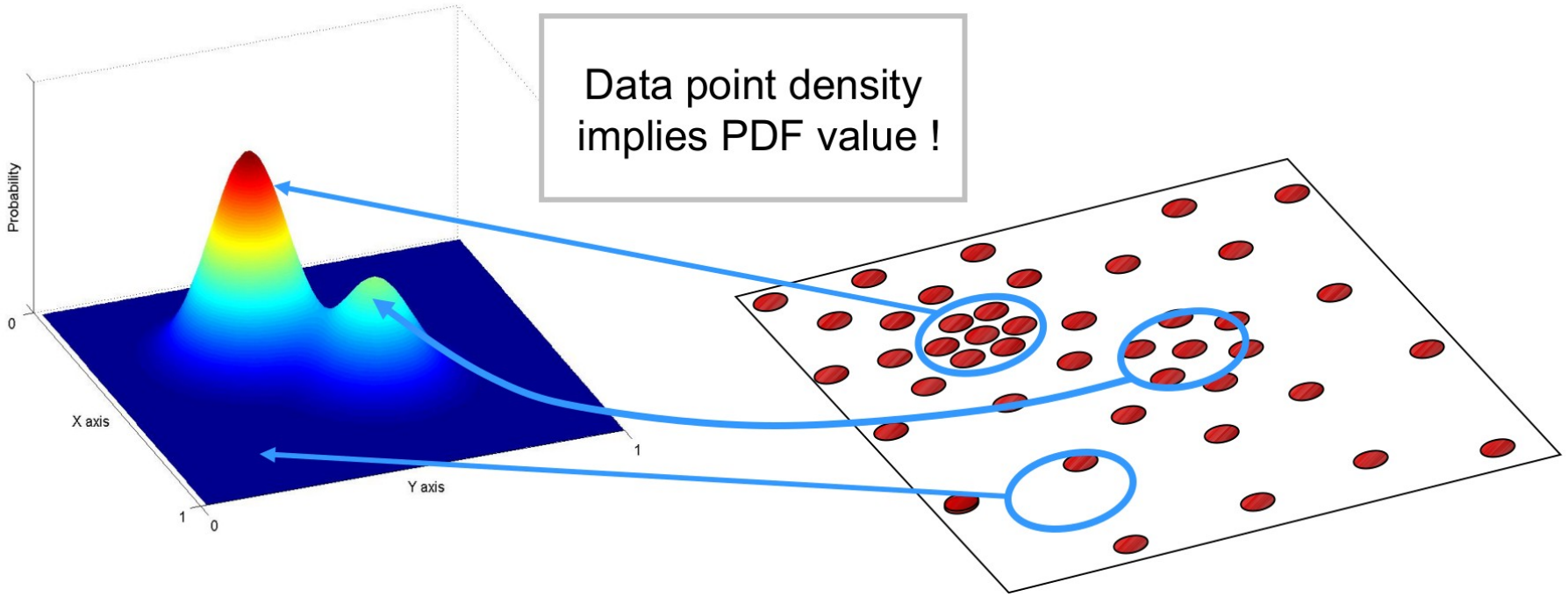
Mean-shift

Results in “ an edge preserving smooth image”

Courtesy: Christian Schellewald, 2020

Non-Parametric Density Estimation

Assumption: The data points are sampled from an underlying PDF



Assumed Underlying PDF
(Probability Density Function)

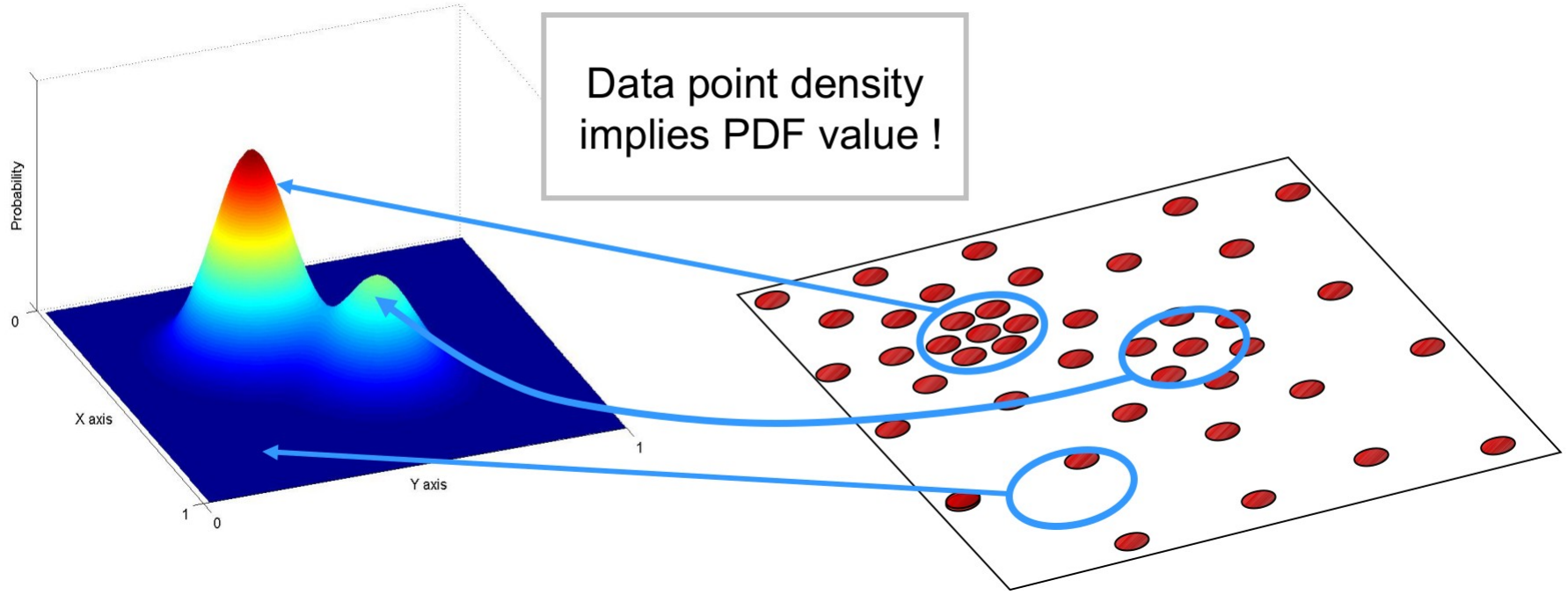
Real Data Samples

Courtesy: Christian Schellewald, 2020

Parametric Density Estimation

Assumption : The data points are sampled from an underlying PDF

Assumption: Probability function is sum of “Bell shaped Gaussians”



Assumed Underlying PDF

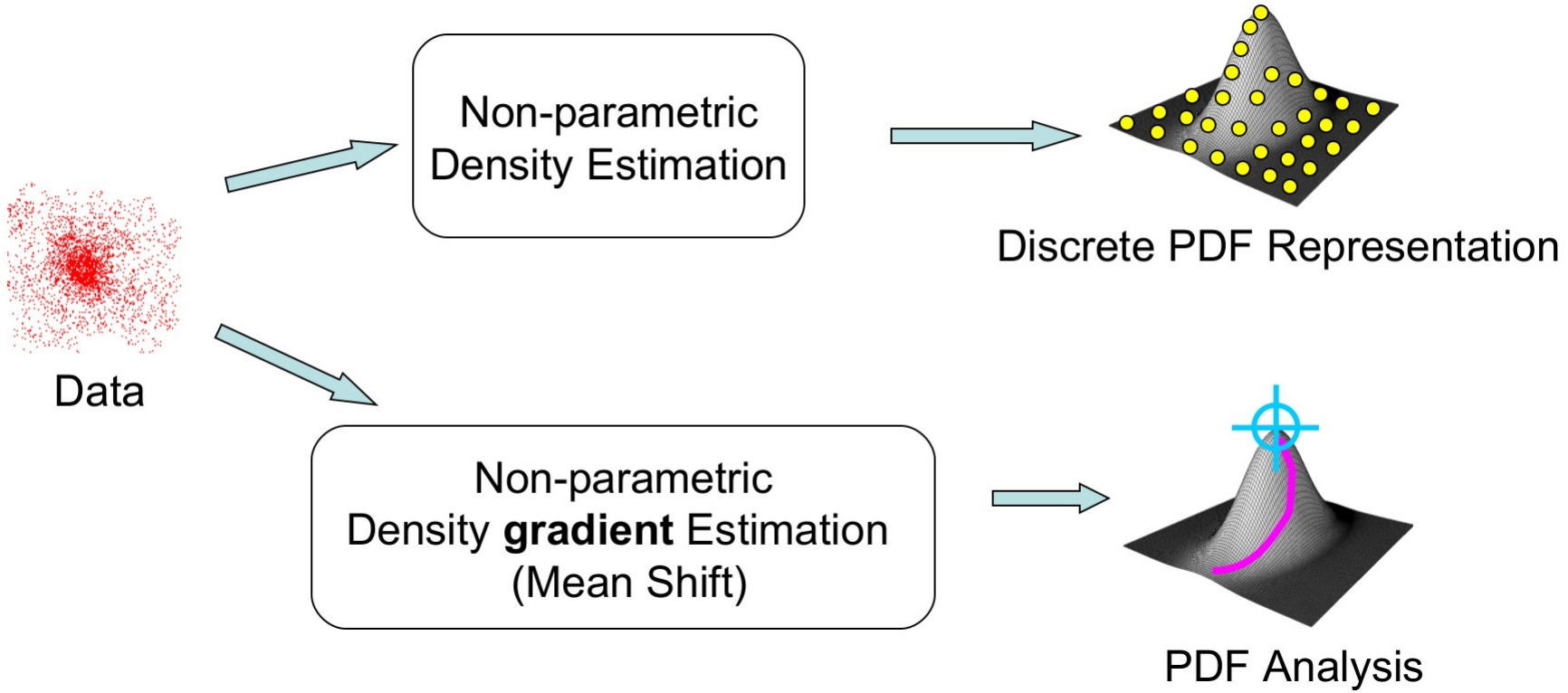
Real Data Samples

Courtesy: Christian Schellewald, 2020

What is Mean-Shift?

A tool for:

Finding modes (=maximal values) in a set of data samples, manifesting an underlying probability density function (PDF) in R N



Courtesy: Christian Schellewald, 2020

Kernel Density Estimation

Parzen Windows - General Framework

$$P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{x} - \mathbf{x}_i)$$

A function of some finite number of data points $x_1 \ldots x_n$

Main Kernel Properties:

Normalized

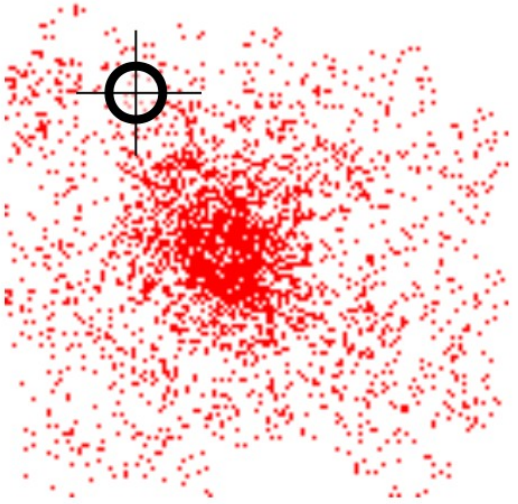
$$\int_{R^d} K(\mathbf{x})d\mathbf{x} = 1$$

Symmetric

$$\int_{R^d} \mathbf{x}K(\mathbf{x})d\mathbf{x} = 0$$

Exponential weight decay

$$\lim_{\|\mathbf{x}\|\to\infty} \|\mathbf{x}\|^d K(\mathbf{x}) = 0$$



Data

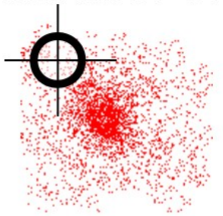
Courtesy: Christian Schellewald, 2020

Kernel Density Estimation

Parzen Windows - Function Forms

$$P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{x} - \mathbf{x}_i)$$

A function of some finite number of data points $x_1 \ldots x_n$



Data

In practice one uses the forms:

$$K(\mathbf{x}) = c\prod_{i=1}^{d} k(x_i) \qquad \text{or} \qquad K(\mathbf{x}) = ck\left(\|\mathbf{x}\|\right)$$

Same function in each dimension

Function of vector length only

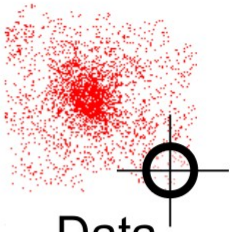
Courtesy: Christian Schellewald, 2020

Kernel Density Estimation

Various Kernels

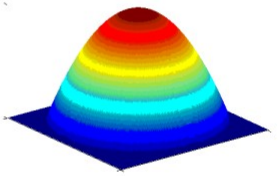
$$P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{x} - \mathbf{x}_i)$$

A function of some finite number of data points $x_1 \ldots x_n$

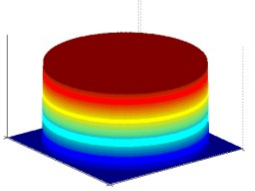


Examples:

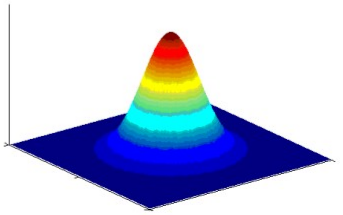
- Epanechnikov Kernel $K_E(\mathbf{x}) = \begin{cases} c\left(1 - \|\mathbf{x}\|^2\right) & \|\mathbf{x}\| \le 1 \\ 0 & \text{otherwise} \end{cases}$



- Uniform Kernel $K_U(\mathbf{x}) = \begin{cases} c & \|\mathbf{x}\| \le 1 \\ 0 & \text{otherwise} \end{cases}$



- Normal Kernel $K_N(\mathbf{x}) = c \times \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)$



Courtesy: Christian Schellewald, 2020

Kernel Density Gradient Estimation

$$\nabla P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla K(\mathbf{x} - \mathbf{x}_i)$$

Give up estimating the PDF !
Estimate **ONLY** the gradient

Using the Kernel form:

$$K(\mathbf{x} - \mathbf{x}_i) = ck\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right)$$

Size of window

We get :

$$\nabla P(\mathbf{x}) = \frac{c}{n}\sum_{i=1}^{n} \nabla k_i = \frac{c}{n}\left[\sum_{i=1}^{n} g_i\right] \mathrm{g}\left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g_i}{\sum_{i=1}^{n} g_i} - \mathbf{x}\right]$$

$$\mathrm{g}(\mathbf{x}) = -k'(\mathbf{x})$$

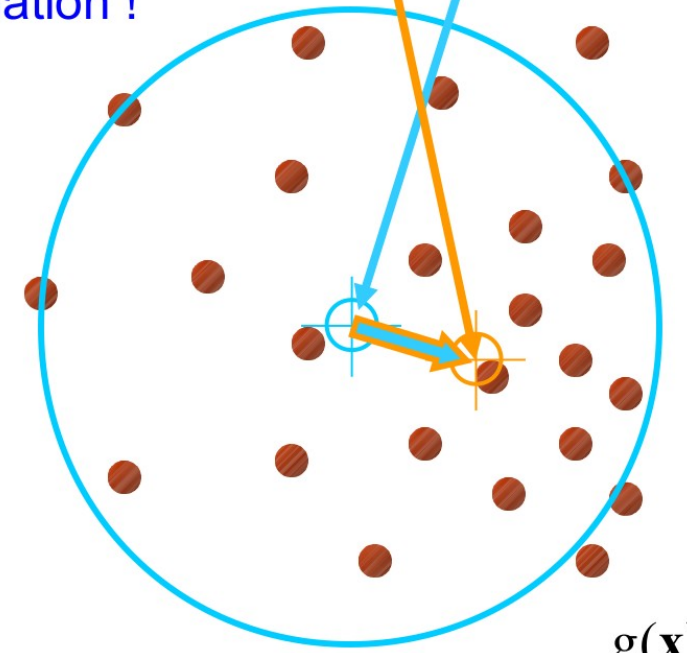
Courtesy: Christian Schellewald, 2020

Computing The Mean Shift

$$\nabla P(\mathbf{x}) = \frac{c}{n}\sum_{i=1}^{n}\nabla k_i = \frac{c}{n}\left[\sum_{i=1}^{n} g_i\right]\mathrm{g}\left[\frac{\sum_{i=1}^{n}\mathbf{x}_i g_i}{\sum_{i=1}^{n} g_i} - \mathbf{x}\right]$$

Gradient at current mode-estimate

A Kernel density estimation !



$$\mathrm{g}(\mathbf{x}) = -k'(\mathbf{x})$$

Simple Mean Shift procedure:

- Compute mean shift vector

$$\mathbf{m}(\mathbf{x}) = \left[\frac{\sum_{i=1}^{n}\mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)} - \mathbf{x}\right]$$

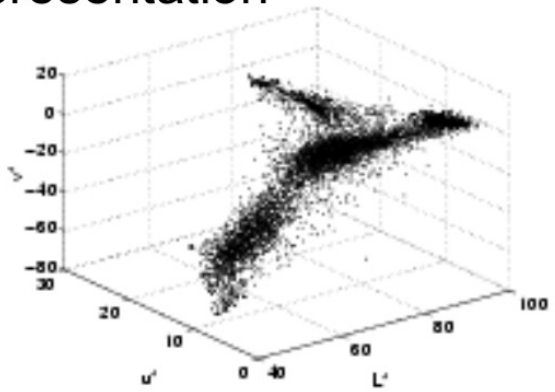
- Translate the Kernel window by **m(x)**

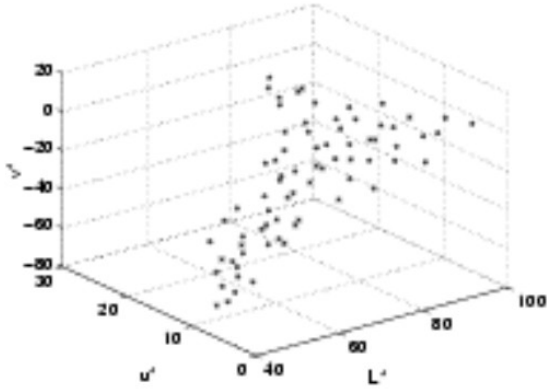
Courtesy: Christian Schellewald, 2020

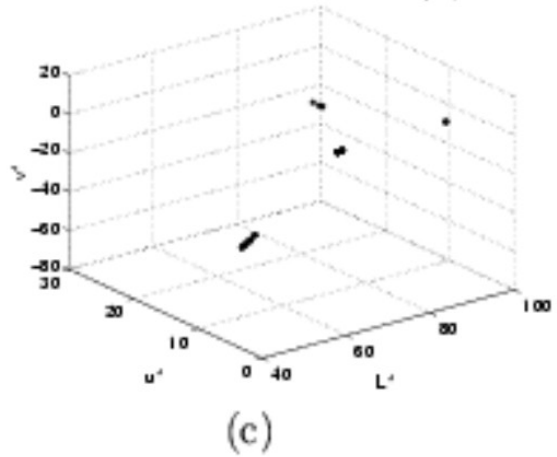
Clustering Real Example

Feature space:
L*u*v representation

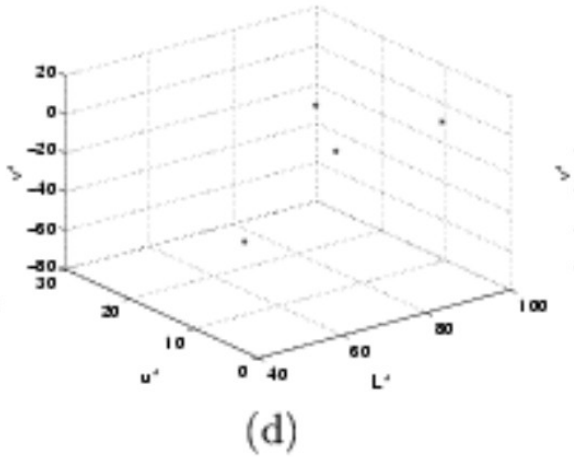
Initial window centers



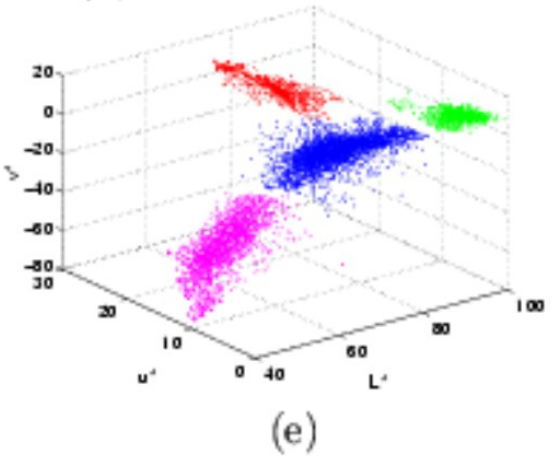




Modes found

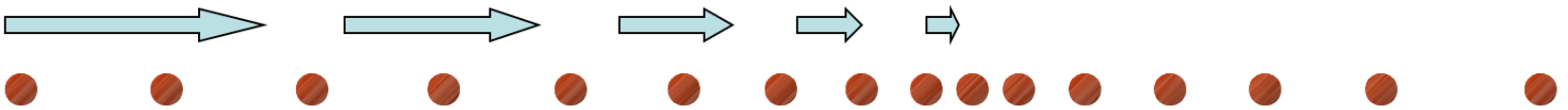


Modes after pruning



Final clusters

Mean Shift Properties



- Automatic convergence speed – the mean shift vector size depends on the gradient itself.
- Near maxima, the steps are small and refined

Adaptive Gradient Ascent

- Convergence is guaranteed for infinitesimal steps only ➔ infinitely convergent, (therefore set a lower bound/threshold as stop criteria)
- For Uniform Kernel ( ), convergence is achieved in a finite number of steps
- Normal Kernel ( ) exhibits a smooth trajectory, but is slower than Uniform Kernel ( ).

Courtesy: Christian Schellewald, 2020

Mean Shift Strengths & Weaknesses

Strengths:

- Application independent tool
- Suitable for real data analysis
- Does not assume any prior shape (e.g. elliptical) on data clusters
- Can handle arbitrary feature spaces
- Only ONE parameter to choose
- h (window size) has a physical Meaning

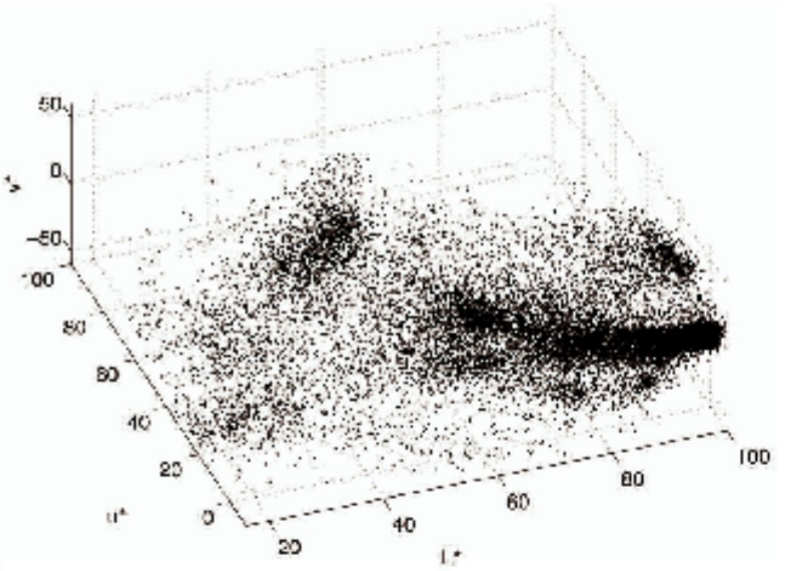
Weaknesses:

- The window size (bandwidth selection) is not trivial
- Inappropriate window size can cause modes to be merged, or generate additional “shallow” modes → Use adaptive window size

Courtesy: Christian Schellewald, 2020

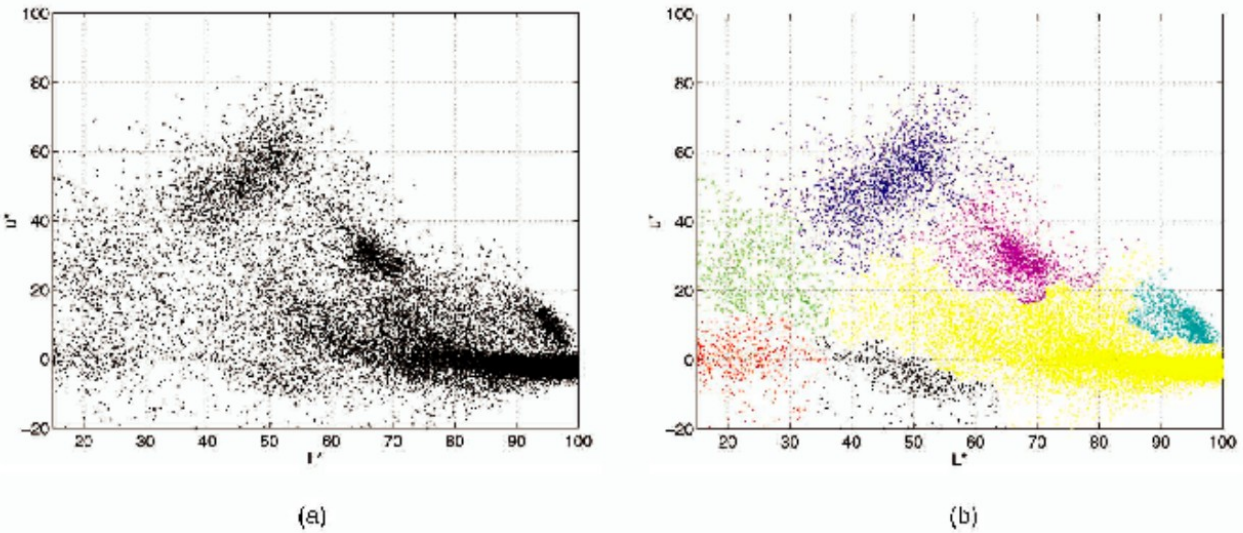
Clustering - Real Example

L*u*v color space representation

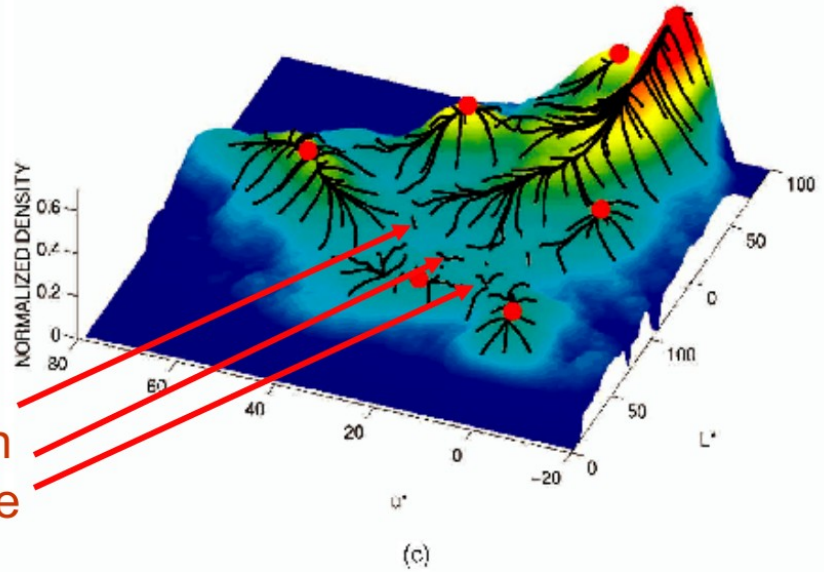


Clustering - Real Example

2D (L*u) space representation



Final clusters



Not all trajectories in the attraction basin reach the same mode

Support Vector Machine (SVM)

SVMs have been used by for both face detection and face recognition and are a widely used tool in object recognition in general.

Basic idea of support vector machines:

- Just like 1-layer or multi-layer neural nets
- Optimal **hyperplane** for linearly separable patterns
- **Extend** to patterns that are **not linearly**
- **separable by transformations** of original data to map into new space – the **Kernel function**
- **Support vectors** are the data points that lie closest to the decision surface (or hyperplane)
  - They are the data points most difficult to classify
  - They have direct bearing on the optimum location of the decision surface
  - We can show that the optimal hyperplane stems from the function class with the lowest“capacity”= # of independent features/parameters we can twiddle

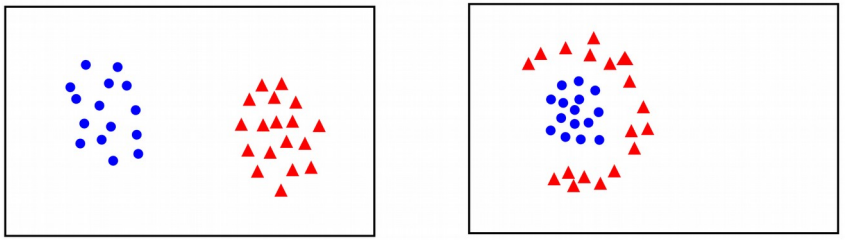
The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

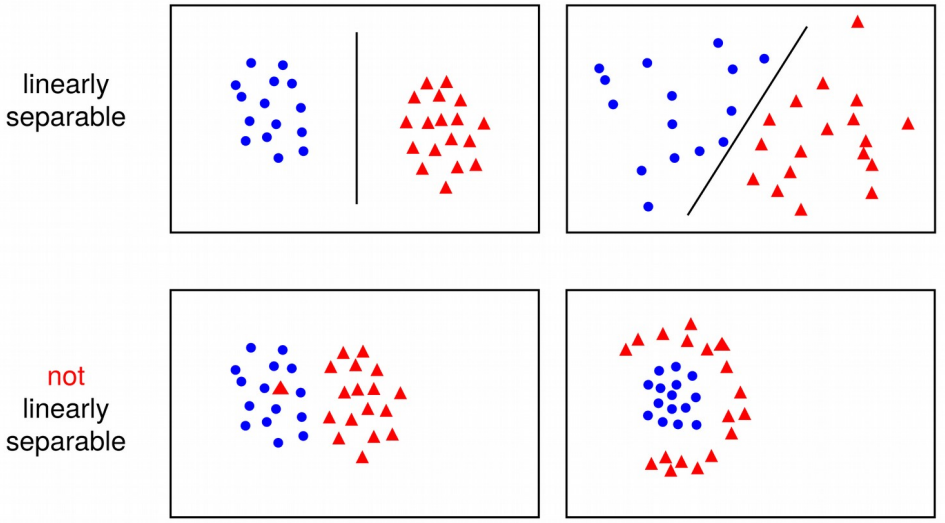
Binary Classification

Given training data $(\mathbf{x}_i, y_i)$ for $i = 1 \ldots N$, with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$, learn a classifier $f(\mathbf{x})$ such that

$$f(\mathbf{x}_i) \begin{cases} \geq 0 & y_i = +1 \\ < 0 & y_i = -1 \end{cases}$$

i.e. $y_i f(\mathbf{x}_i) > 0$ for a correct classification.



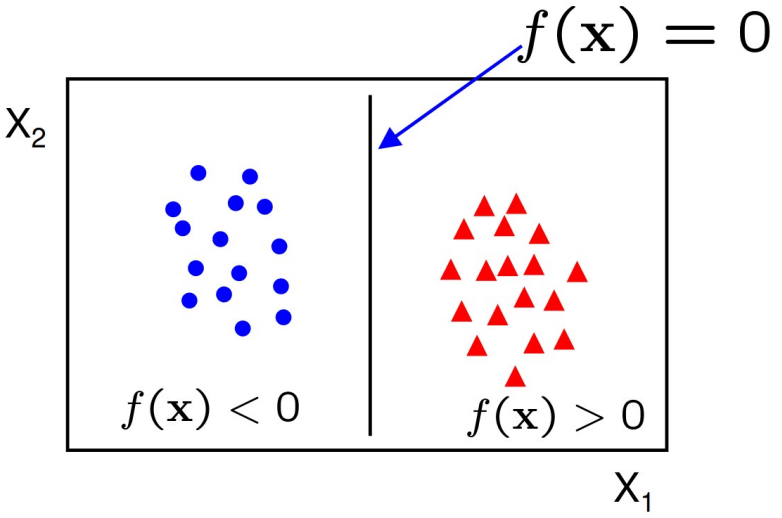


The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Linear Classifiers

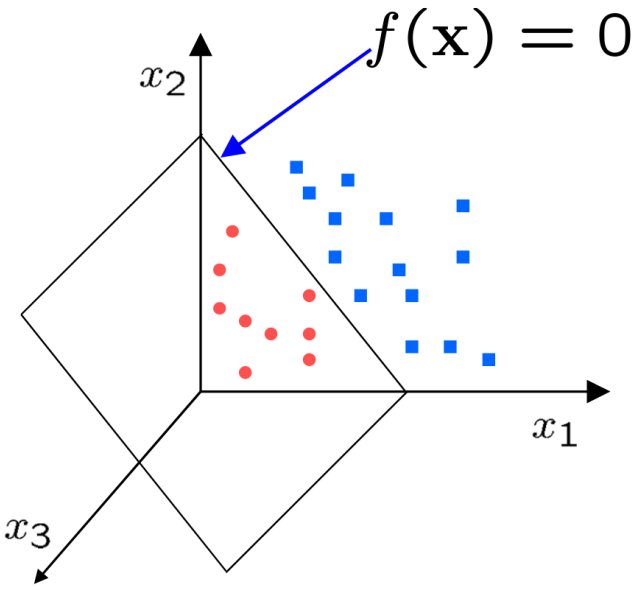
A linear classifier has the form

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$



- in 2D the discriminant is a line
- $\mathbf{w}$ is the normal to the line, and b the bias
- $\mathbf{w}$ is known as the weight vector

- in 3D the discriminant is a plane, and in nD it is a hyperplane



The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Linear Classifiers

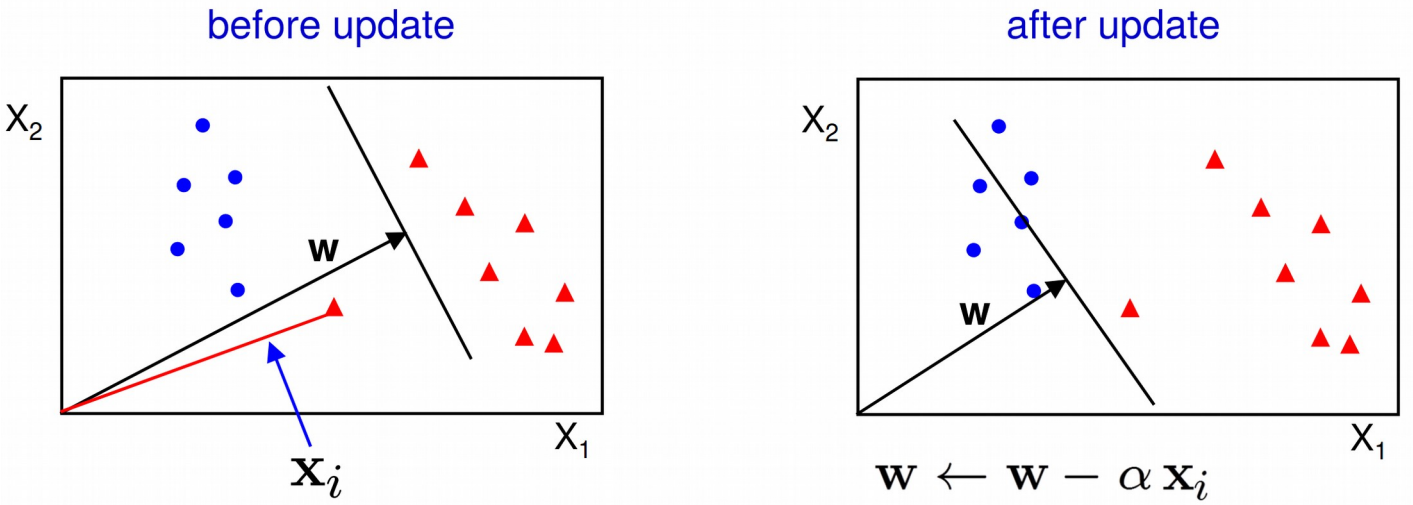
For a K-NN classifier it was necessary to `carry' the training data

For a linear classifier, the training data is used to learn **w** and then discarded

Only **w** is needed for classifying new data

2D Example:

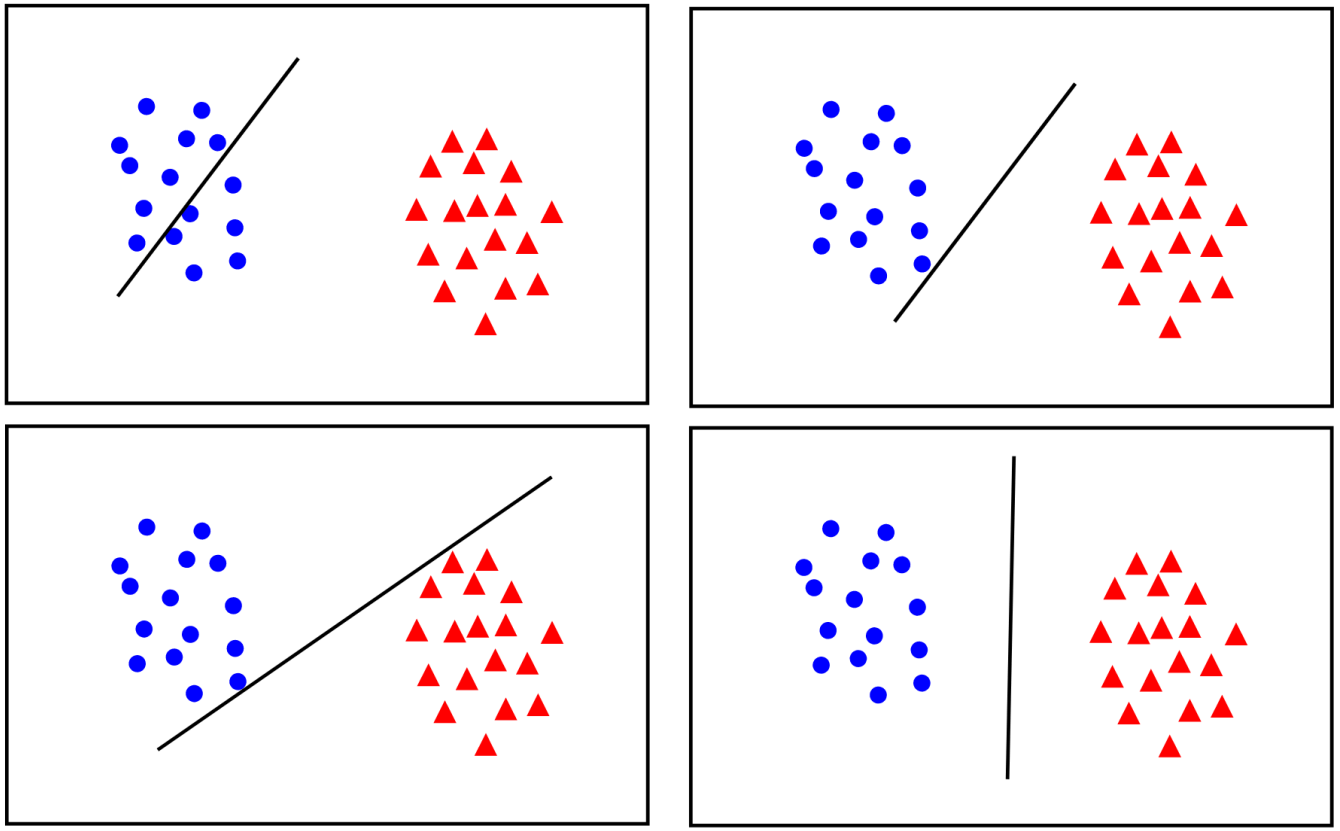
- Initialize **w** = 0
- Cycle though the data points { $\mathbf{x}_i$, y_i }
  - if $\mathbf{x}_i$ is misclassified then $\mathbf{w} \leftarrow \mathbf{w} + \alpha \, \text{sign}(f(\mathbf{x}_i)) \, \mathbf{x}_i$
- Until all the data is correctly classified



The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Linear Classifers

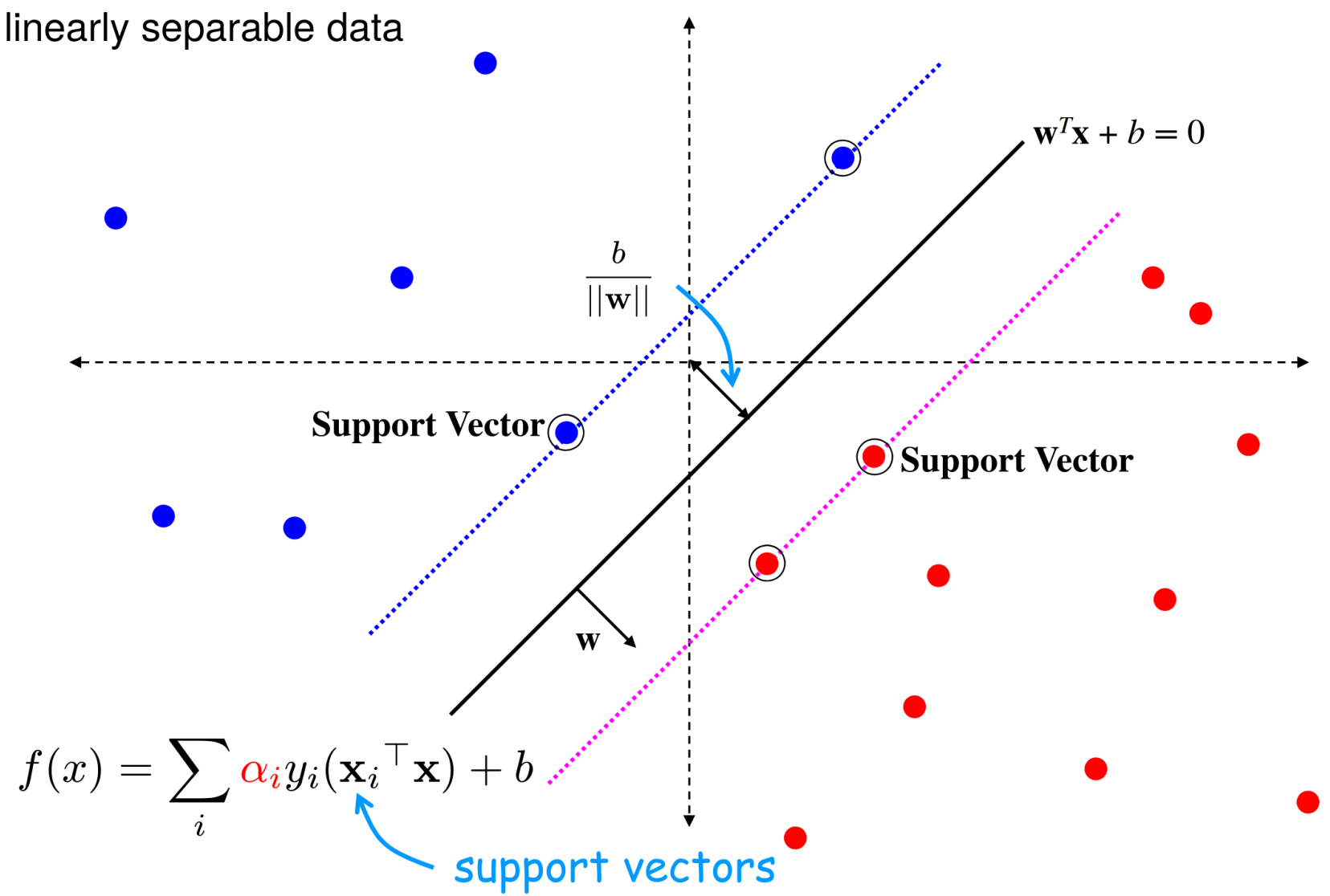
- if the data is linearly separable, then the algorithm will converge
- Convergence can be slow
- separating line close to training data
- we would prefer a larger margin for generalization
- maximum margin solution: most stable under perturbations of the inputs



What is the best **w** ?

The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Support Vector Machine



The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Support Vector Machine

- Since $\mathbf{w}^\top \mathbf{x} + b = 0$ and $c(\mathbf{w}^\top \mathbf{x} + b) = 0$ define the same plane, we have the freedom to choose the normalization of $\mathbf{w}$

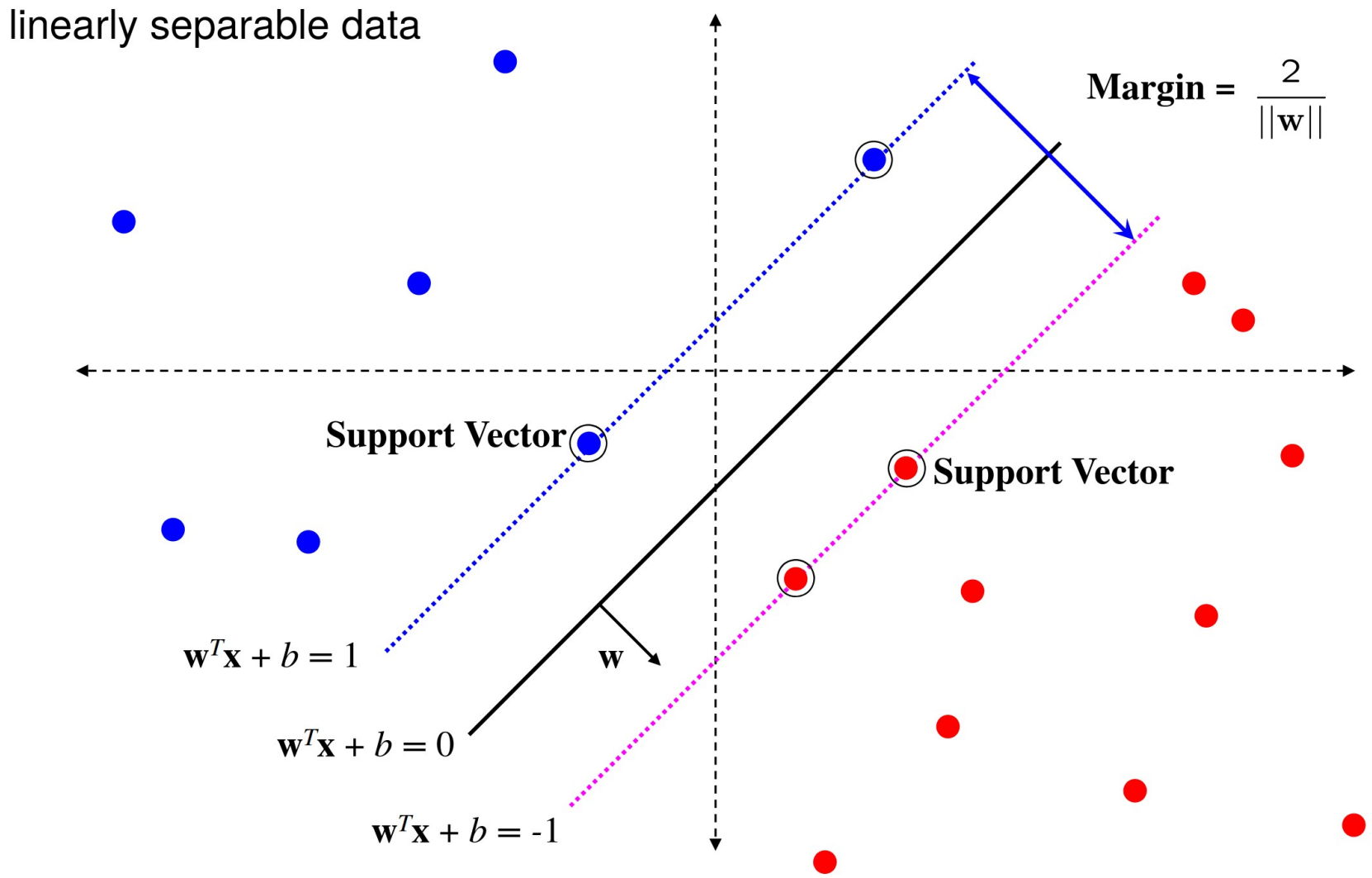
- Choose normalization such that $\mathbf{w}^\top \mathbf{x}_+ + b = +1$ and $\mathbf{w}^\top \mathbf{x}_- + b = -1$ for the positive and negative support vectors respectively

- Then the margin is given by

$$\frac{\mathbf{w}}{||\mathbf{w}||} \cdot \left(\mathbf{x}_+ - \mathbf{x}_-\right) = \frac{\mathbf{w}^\top \left(\mathbf{x}_+ - \mathbf{x}_-\right)}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||}$$

The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Support Vector Machine



The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Support Vector Machine

- Learning the SVM can be formulated as an optimization:

$$\max_{\mathbf{w}} \frac{2}{||\mathbf{w}||} \text{ subject to } \mathbf{w}^\top \mathbf{x}_i + b \begin{array}{ll} \geq 1 & \text{if } y_i = +1 \\ \leq -1 & \text{if } y_i = -1 \end{array} \text{ for } i = 1 \ldots N$$

- Or equivalently

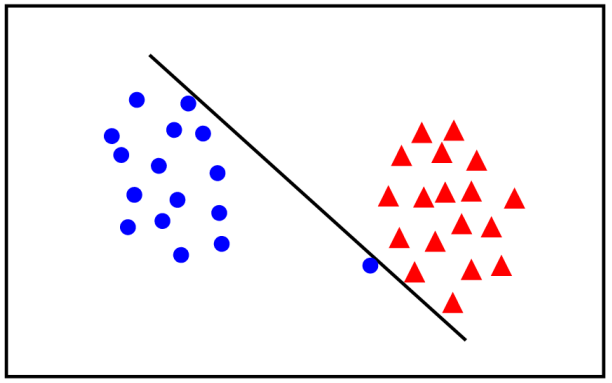
$$\min_{\mathbf{w}} ||\mathbf{w}||^2 \text{ subject to } y_i \left(\mathbf{w}^\top \mathbf{x}_i + b \right) \geq 1 \text{ for } i = 1 \ldots N$$

- This is a quadratic optimization problem subject to linear constraints and there is a unique minimum

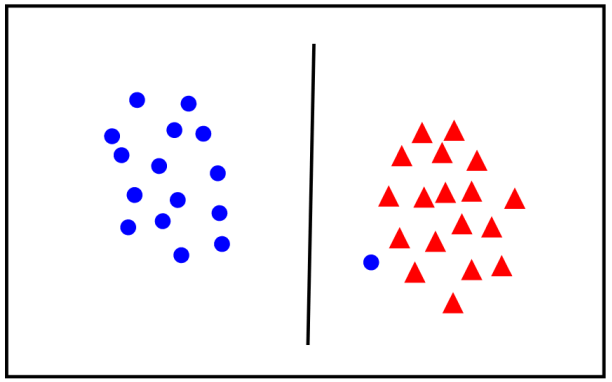
The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Support Vector Machine

Linear separability



• the points can be linearly separated but there is a very narrow margin



• but possibly the large margin solution is better, even though one constraint is violated

In general there is a trade off between the margin and the number of mistakes on the training data

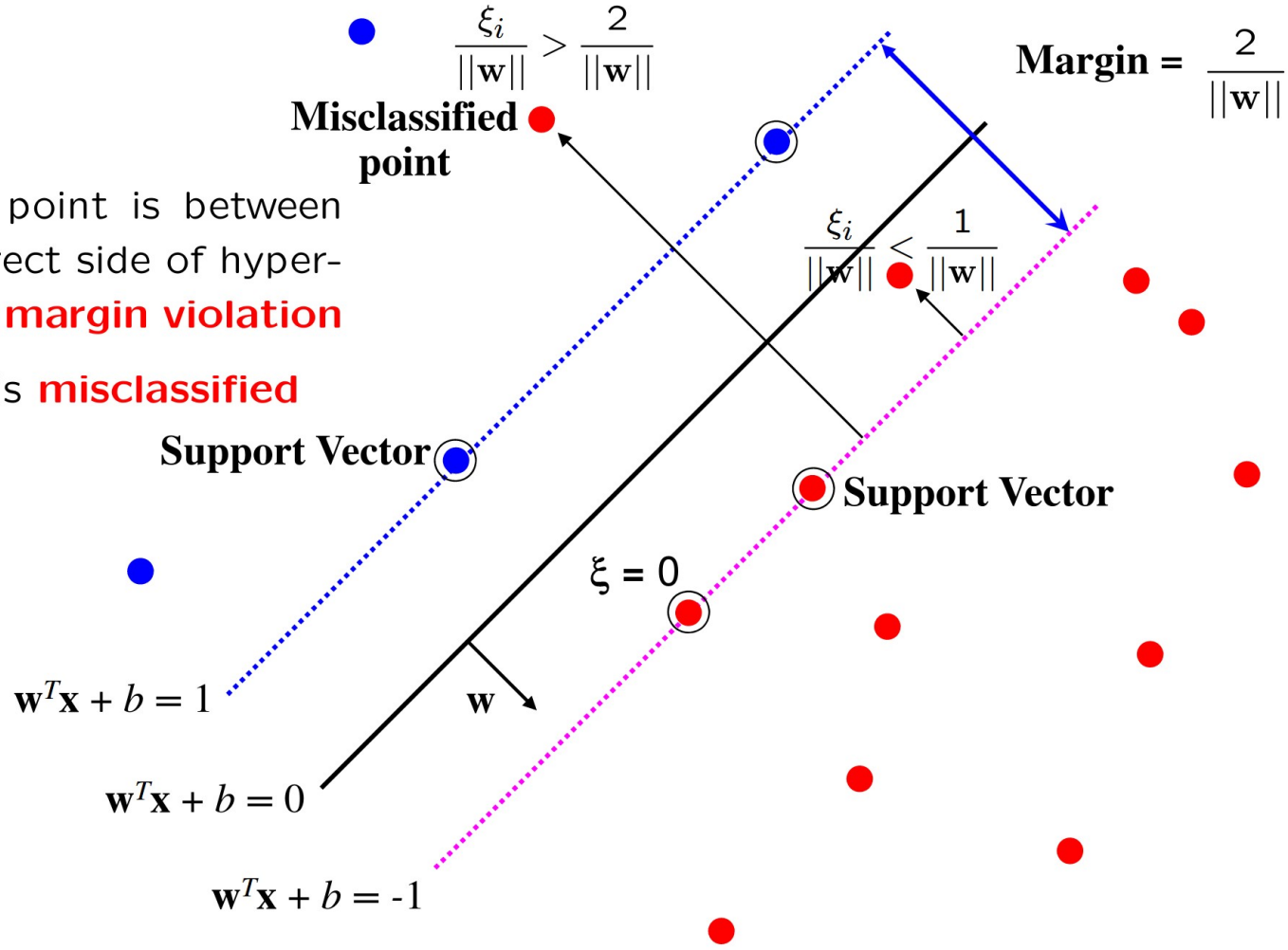
The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Support Vector Machine

Introduce “slack” variables

$$\xi_i \geq 0$$

- for $0 < \xi \leq 1$ point is between margin and correct side of hyper-plane. This is a **margin violation**
- for $\xi > 1$ point is **misclassified**



The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Support Vector Machine

"Soft" margin solution: Data is linearly separable but only with a narrow margin

The optimization problem becomes

$$\min_{\mathbf{w}\in\mathbb{R}^d,\xi_i\in\mathbb{R}+} ||\mathbf{w}||^2+C\sum_i^N \xi_i$$

subject to

$$y_i\left(\mathbf{w}^\top\mathbf{x}_i+b\right)\geq 1-\xi_i \text{ for } i=1\ldots N$$

- Every constraint can be satisfied if ξ_i is sufficiently large
- C is a regularization parameter:
  - small C allows constraints to be easily ignored $\rightarrow$ large margin
  - large C makes constraints hard to ignore $\rightarrow$ narrow margin
  - $C=\infty$ enforces all constraints: hard margin
- This is still a quadratic optimization problem and there is a unique minimum. Note, there is only one parameter, C.

The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Support Vector Machine

Application: Pedestrian detection

Objective: detect (localize) standing humans in an image

- cf face detection with a sliding window classifier

- reduces object detection to binary classification

- does an image window contain a person or not?

Method: the HOG detector

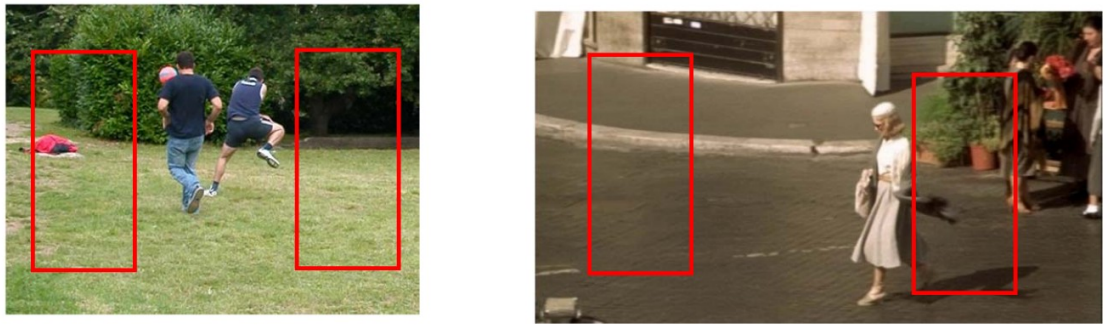
The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Support Vector Machine

Training data and features

- Positive data – 1208 positive window examples

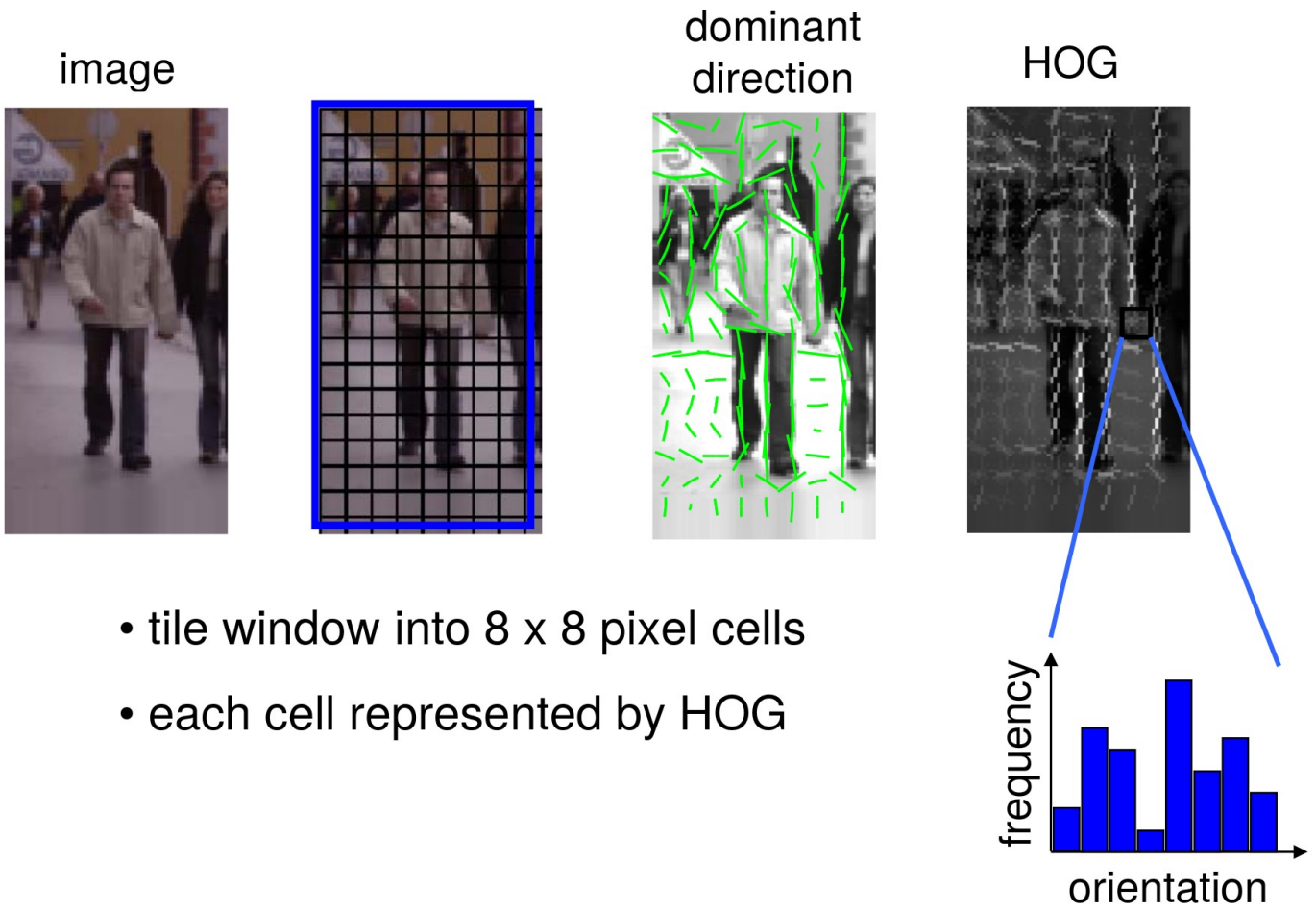
- Negative data – 1218 negative window examples (initially)



The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Support Vector Machine

Feature: histogram of oriented gradients (HOG)



- tile window into 8 x 8 pixel cells
- each cell represented by HOG

Feature vector dimension = 16 x 8 (for tiling) x 8 (orientations) = 1024

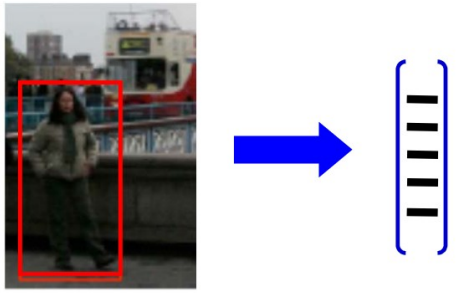
The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Support Vector Machine

Algorithm

Training (Learning)

- Represent each example window by a HOG feature vector



$$\mathbf{x}_i \in \mathbb{R}^d, \text{ with } d = 1024$$

- Train a SVM classifier

Testing (Detection)

- Sliding window classifier

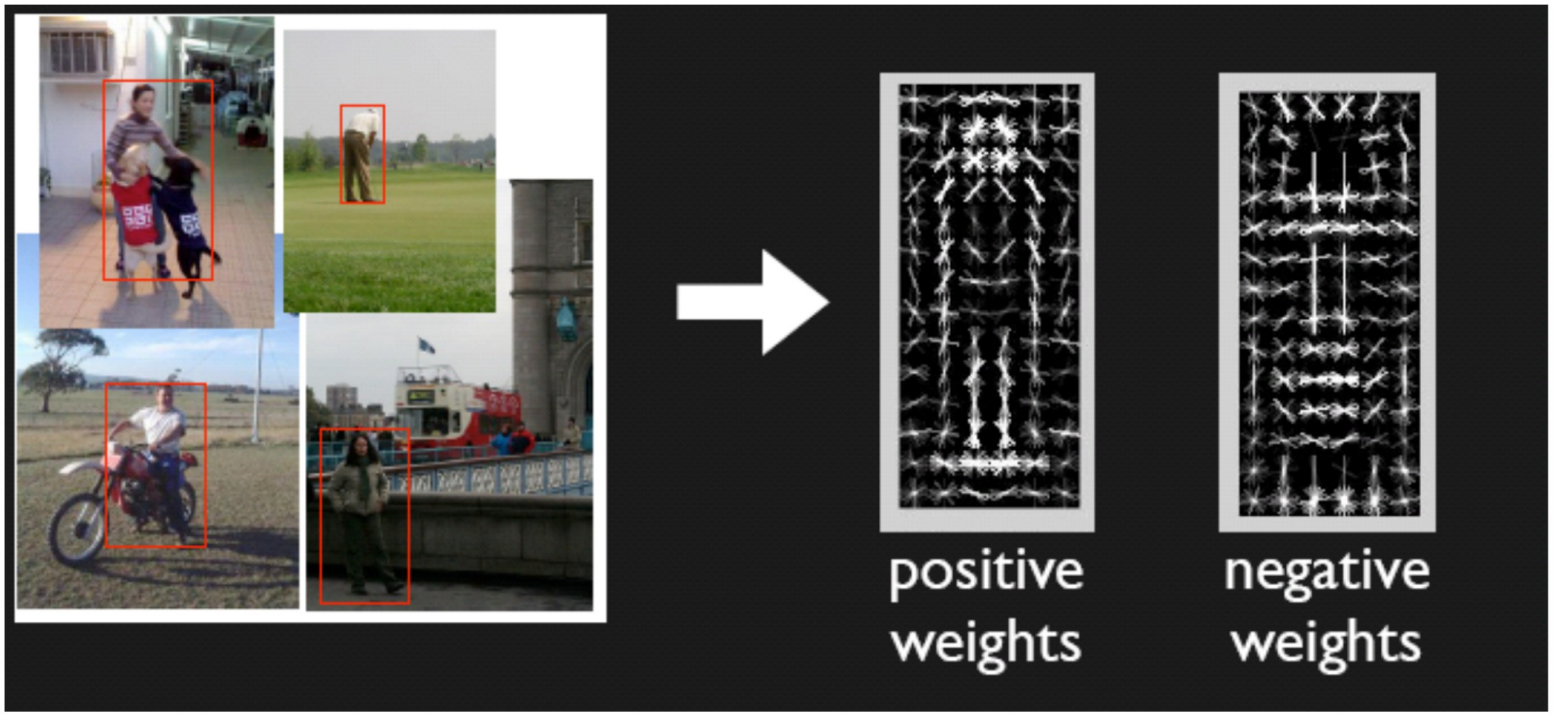
$$f(x) = \mathbf{w}^\top \mathbf{x} + b$$

The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Support Vector Machine

Learned model

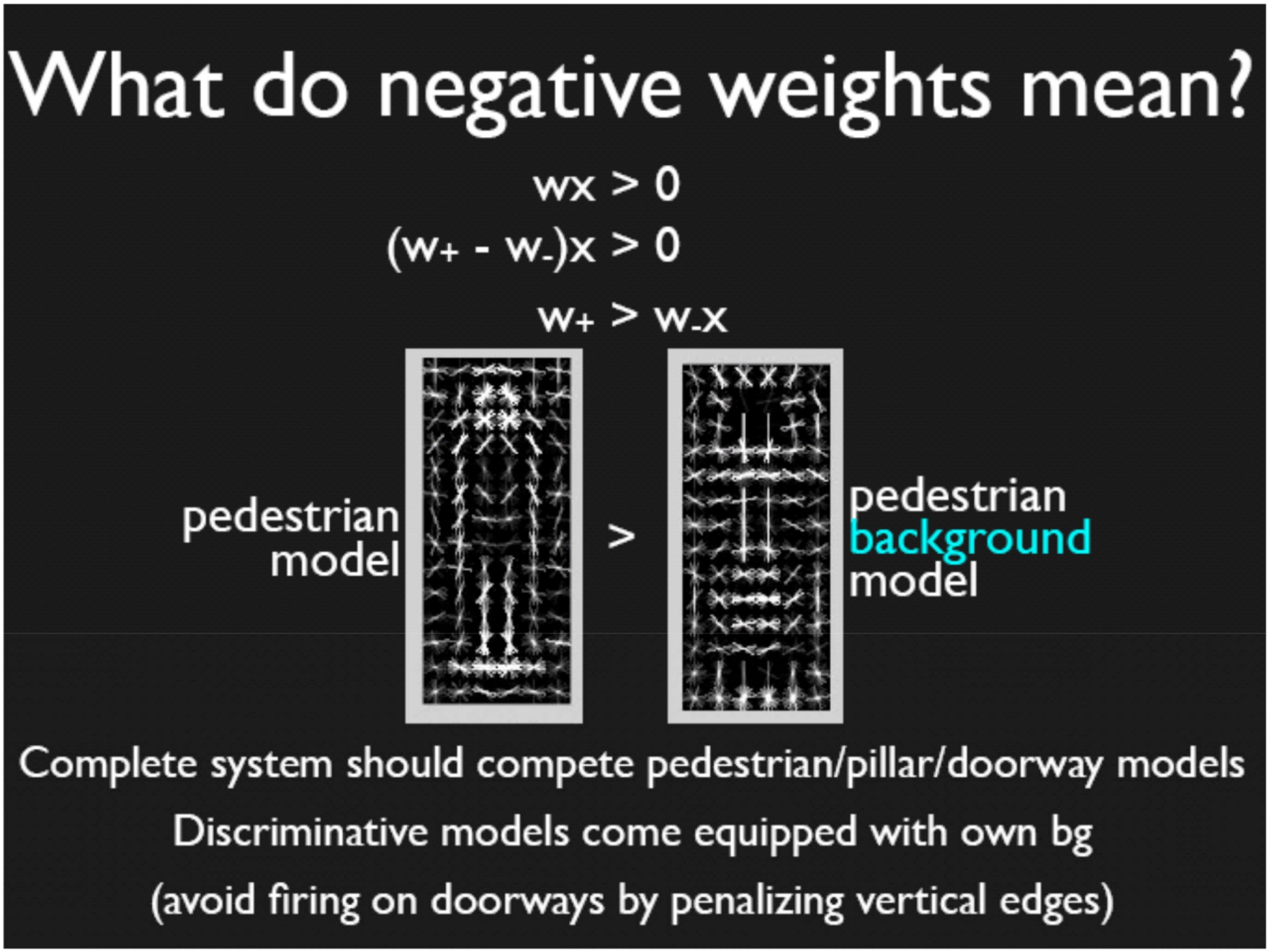
$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$



Slide from Deva Ramanan

The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Support Vector Machine



Slide from Deva Ramanan

The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Support Vector Machine

Learning an SVM has been formulated as a constrained optimization problem over $\mathbf{w}$ and ξ

$$\min_{\mathbf{w}\in\mathbb{R}^d,\xi_i\in\mathbb{R}+} ||\mathbf{w}||^2 + C\sum_i^N \xi_i \text{ subject to } y_i\left(\mathbf{w}^\top\mathbf{x}_i + b\right) \geq 1-\xi_i \text{ for } i = 1\ldots N$$

The constraint $y_i\left(\mathbf{w}^\top\mathbf{x}_i + b\right) \geq 1-\xi_i$, can be written more concisely as

$$y_i f(\mathbf{x}_i) \geq 1-\xi_i$$

which, together with $\xi_i \geq 0$, is equivalent to

$$\xi_i = \max\left(0, 1-y_i f(\mathbf{x}_i)\right)$$

Hence the learning problem is equivalent to the unconstrained optimization problem over $\mathbf{w}$

$$\min_{\mathbf{w}\in\mathbb{R}^d} \underbrace{||\mathbf{w}||^2}_{\text{regularization}} + C\sum_i^N \underbrace{\max\left(0, 1-y_i f(\mathbf{x}_i)\right)}_{\text{loss function}}$$

The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

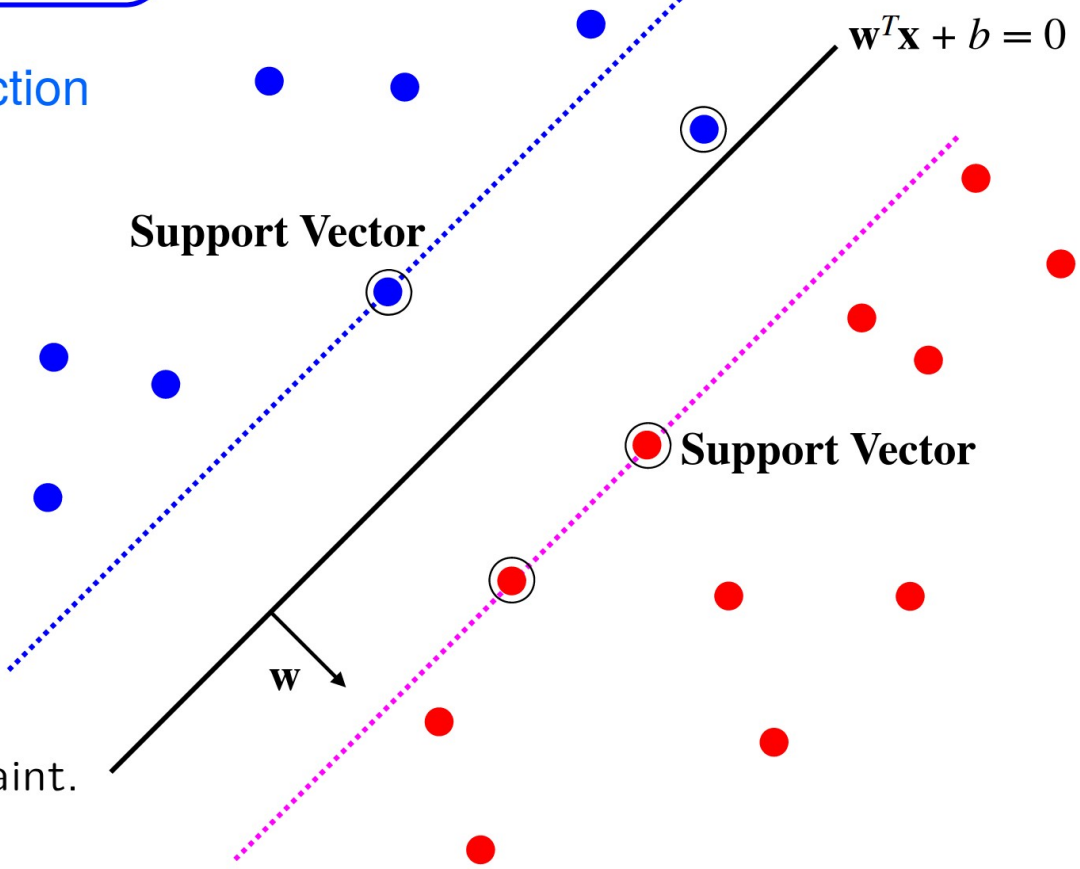
Support Vector Machine

Loss function

$$\min_{\mathbf{w}\in\mathbb{R}^d} ||\mathbf{w}||^2 + C\sum_i^N \underbrace{\max(0, 1 - y_i f(\mathbf{x}_i))}_{\text{loss function}}$$

Points are in three categories:

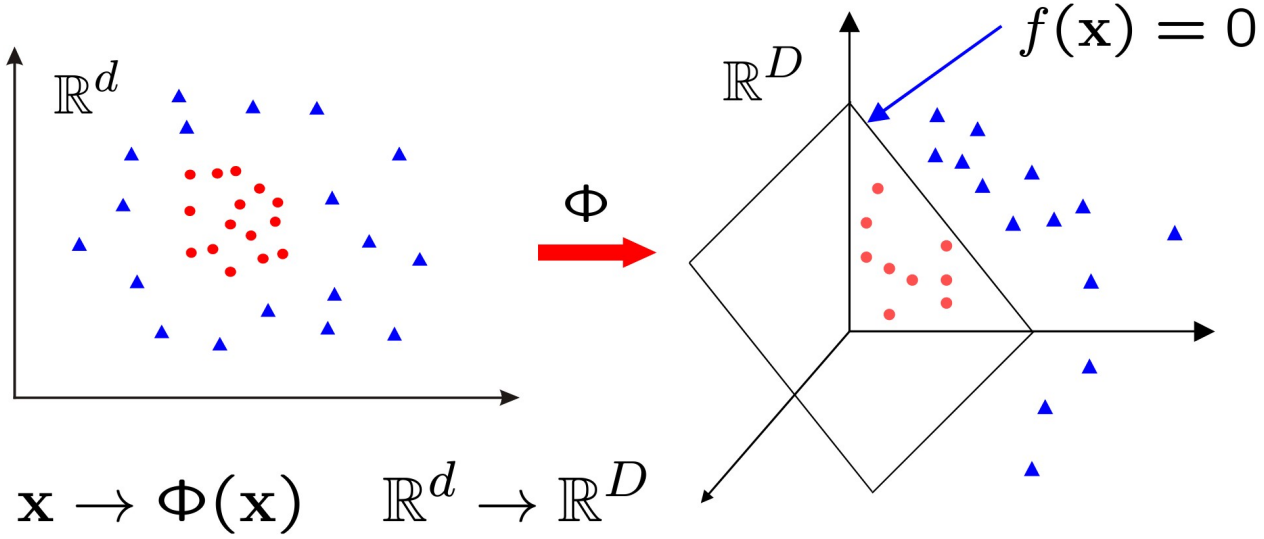
1. $y_i f(x_i) > 1$
   Point is outside margin.
   No contribution to loss
2. $y_i f(x_i) = 1$
   Point is on margin.
   No contribution to loss.
   As in hard margin case.
3. $y_i f(x_i) < 1$
   Point violates margin constraint.
   Contributes to loss



The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Support Vector Machine

Kernal-trick



$$\Phi : \mathbf{x} \rightarrow \Phi(\mathbf{x}) \quad \mathbb{R}^d \rightarrow \mathbb{R}^D$$

Learn classifier linear in $\mathbf{w}$ for $\mathbb{R}^D$:

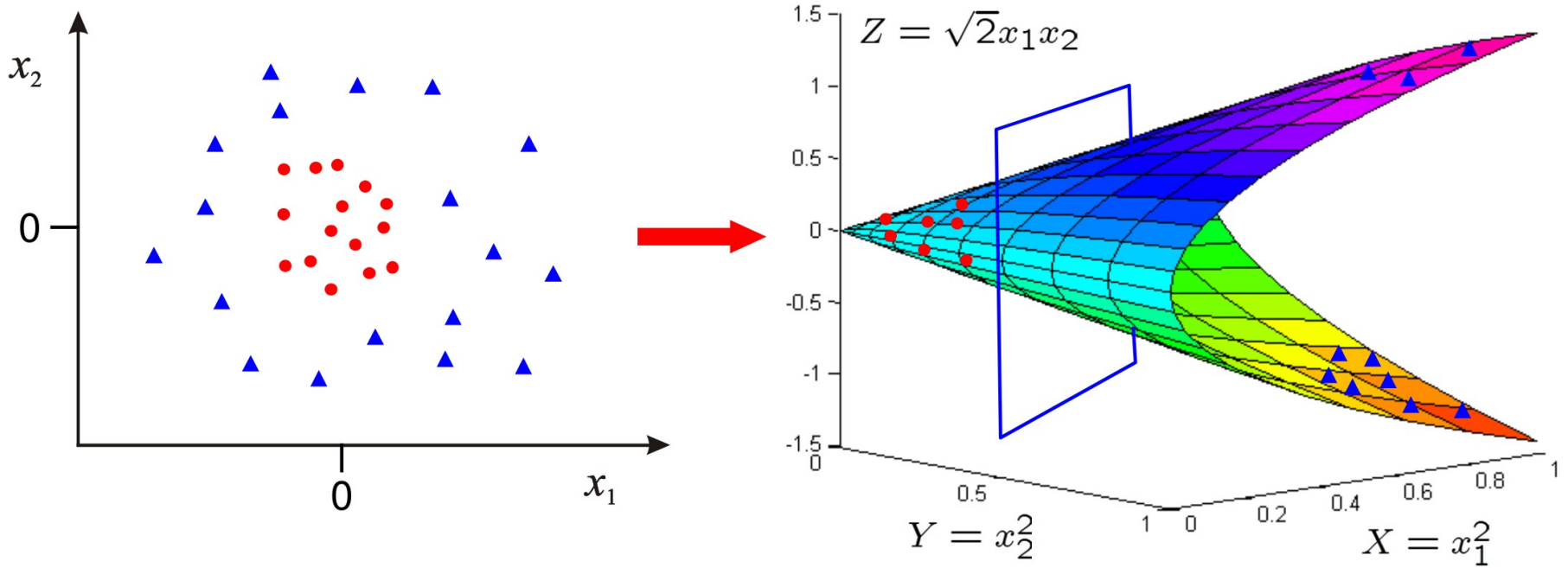
$$f(\mathbf{x}) = \mathbf{w}^\top \Phi(\mathbf{x}) + b$$

$\Phi(\mathbf{x})$ is a feature map

The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Support Vector Machine

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \qquad \mathbb{R}^2 \rightarrow \mathbb{R}^3$$



- Data is linearly separable in 3D
- This means that the problem can still be solved by a linear classifier

The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Support Vector Machine

MATLAB SVM demo

https://www.robots.ox.ac.uk/~az/lectures/ml/

The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

Recognition

Traditional Detection Before Deep Learning

- Feature
  - Haar Feature
  - HOG (Histogram of Gradient)
  - LBP (Local Binary Pattern)
  - ACF (Aggregated Channel Feature)
- Classifier
  - SVM
  - Bootsing
  - Random Forest

Pros

- Efficient to compute (e.g., HAAR, ACF) on CPU
- Easy to debug, analyze the bad cases
- reasonable performance on limited training data

Cons

- Limited performance on large dataset
- Hard to be accelerated by GPU

Detection

How to perform Detection

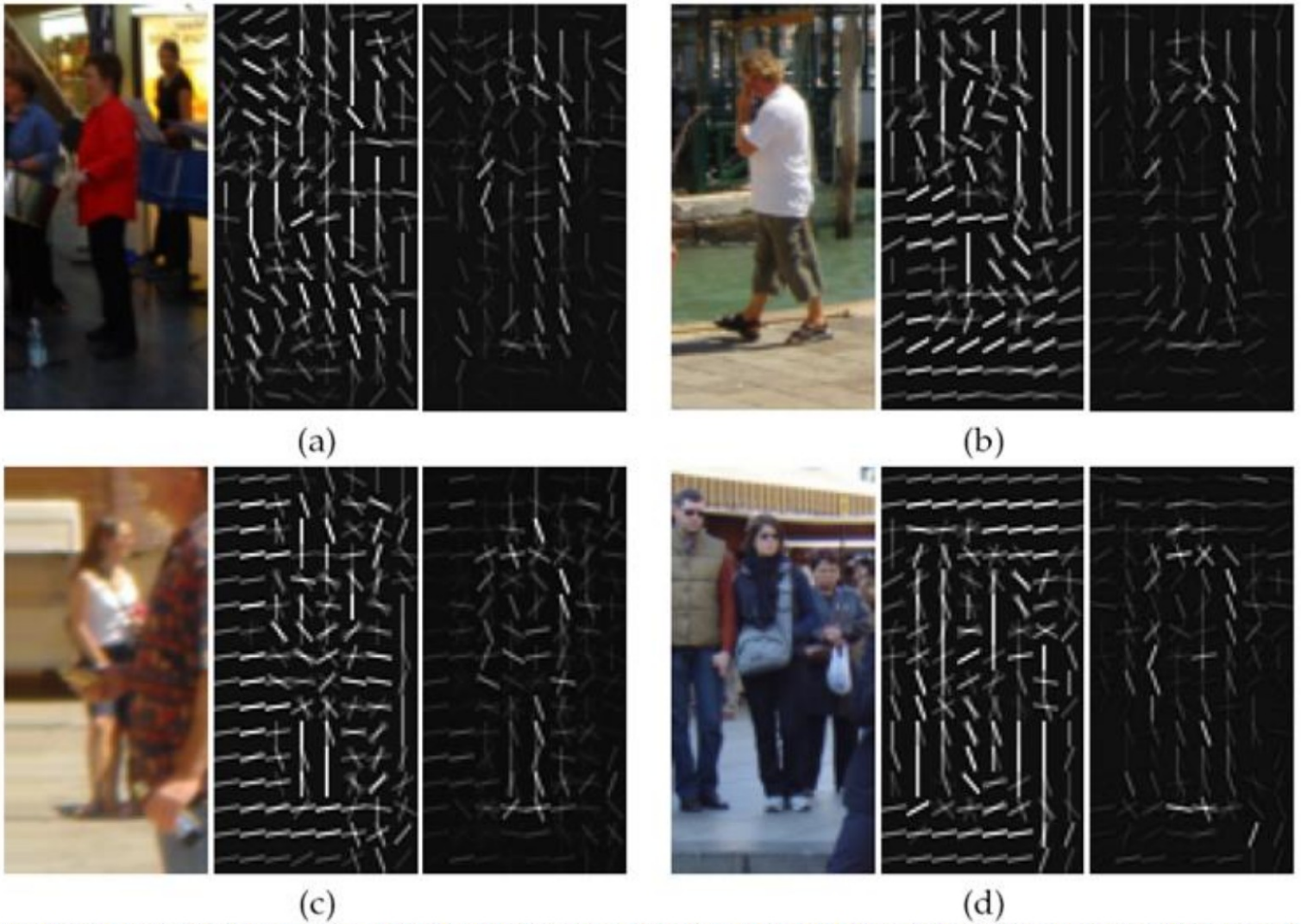
Sliding window: enumerate all the windows (up to millions of windows)

VJ detector: cascade chain

Fully Convolutional network

→ shared computation

Traditional Hand-crafted Feature: HoG



In each triplet: (1) the input image, (2) the corresponding R-HOG feature vector (only the dominant orientation of each cell is shown), (
the dominant orientations selected by the SVM (obtained by multiplying the feature vector by the corresponding weights from the linear SVM).

Deep Learning

$x^{(0)}$ is input feature vector for neural network (one sample).

$x^{(L)}$ is output vector of neural network with L layers.

Layer number l has:

- Inputs (usually $x^{(l-1)}$, i.e. outputs of layer number $l-1$)
- Weight matrix $W^{(l)}$, bias vector $b^{(l)}$ - both trained (e.g. with stochastic gradient descent) such that network output $x^{(L)}$ for the training samples minimizes some objective (loss)
- Nonlinearity s_l (fixed in advance, for example $\mathrm{ReLU}(z) \coloneqq \max\{0, z\}$)
  - Quiz: why is nonlinearity useful?
- Output $x^{(l)}$ of layer l

Transformation from $x^{(l-1)}$ to $x^{(l)}$ performed by layer l:

$$x^{(l)} = s_l\left(W^{(l)}x^{(l-1)} + b^{(l)}\right)$$

Deep Learning. Vladimir Golkov. Technical University of Munich. Computer Vision Group

Deep Learning

$$W^{(l)} = \begin{pmatrix} 0 & 0.1 & -1 \\ -0.2 & 0 & 1 \end{pmatrix}$$

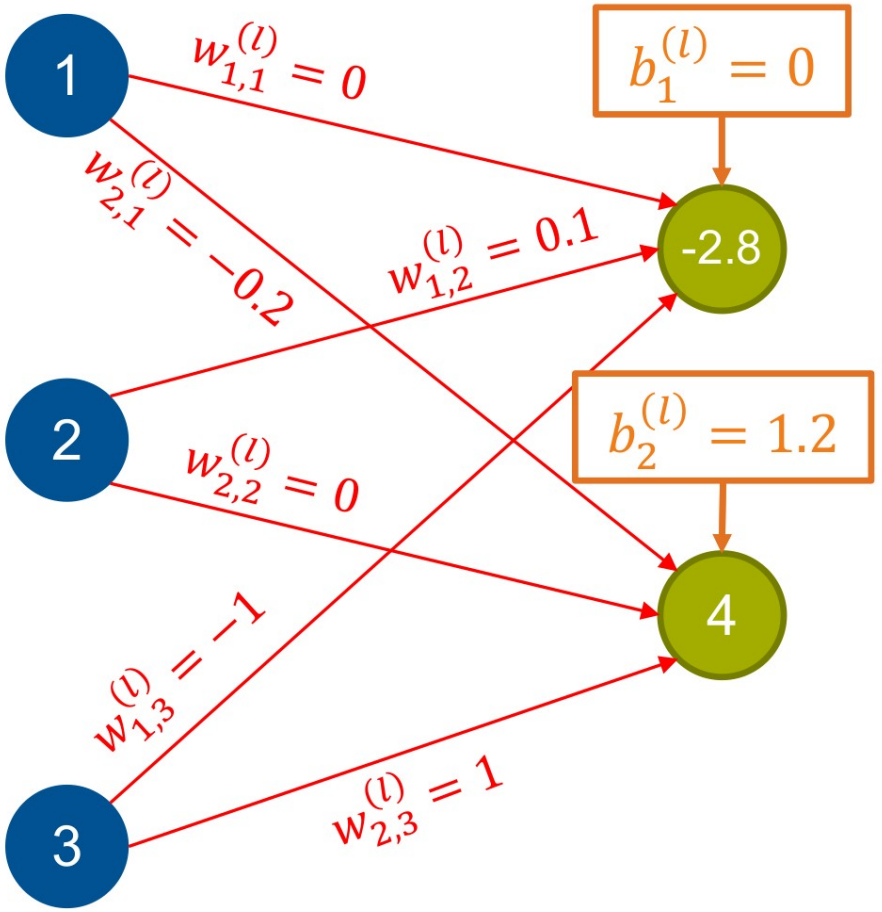
$$x^{(l-1)} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

$$b^{(l)} = \begin{pmatrix} 0 \\ 1.2 \end{pmatrix}$$

$$W^{(l)} x^{(l-1)} + b^{(l)} =$$

$$= \begin{pmatrix} 0 \cdot 1 + 0.1 \cdot 2 - 1 \cdot 3 + 0 \\ -0.2 \cdot 1 + 0 \cdot 2 + 1 \cdot 3 + 1.2 \end{pmatrix}$$

$$= \begin{pmatrix} -2.8 \\ 4 \end{pmatrix}$$

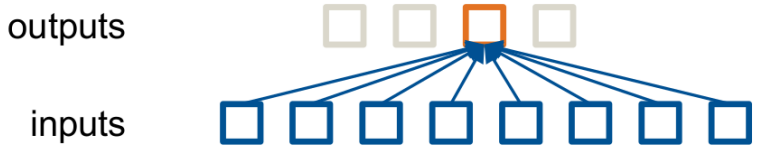


Deep Learning. Vladimir Golkov. Technical University of Munich. Computer Vision Group

Deep Learning

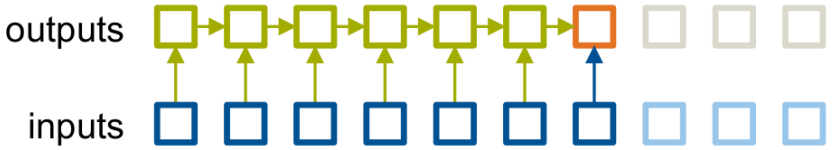
Structured Data

- “Zero-dimensional” data: multilayer perceptron



- Structured data: translation-covariant operations
  - Neighborhood structure: convolutional networks (2D/3D images, 1D bio. sequences, …)

  - Sequential structure (memory): recurrent networks (1D text, 1D audio, …)

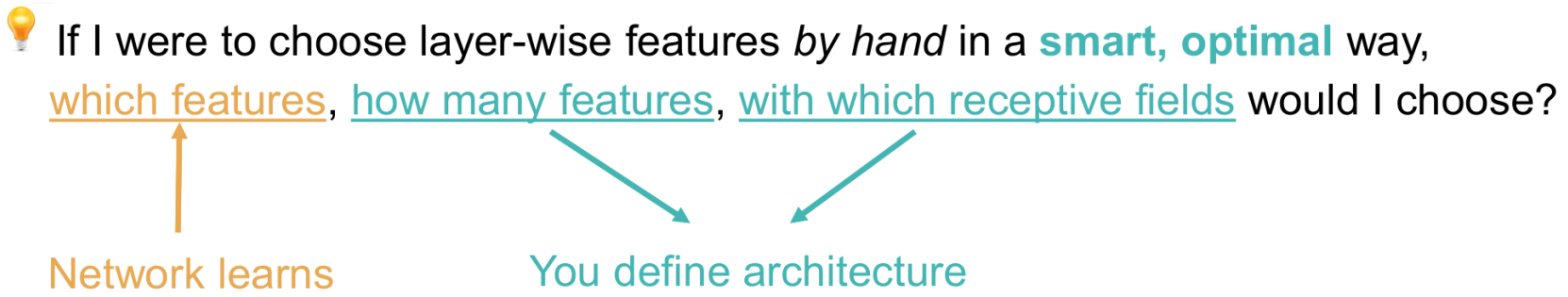


Deep Learning. Vladimir Golkov. Technical University of Munich. Computer Vision Group

Deep Learning

Network Architecture: Your Decision!

Informed approach:



Bottom-up approach: make dataset simple (e.g. simple samples and/or few samples), get a simple network (few layers, few neurons/filters) to work at least on the training set, then re-train increasingly complex networks on increasingly complex data

Top-down approach: use a network architecture that is known to work well on similar data, get it to work on your data, then tune the architecture if necessary

Deep Learning. Vladimir Golkov. Technical University of Munich. Computer Vision Group

Deep Learning – Back Propagation

Why back propagation is needed:

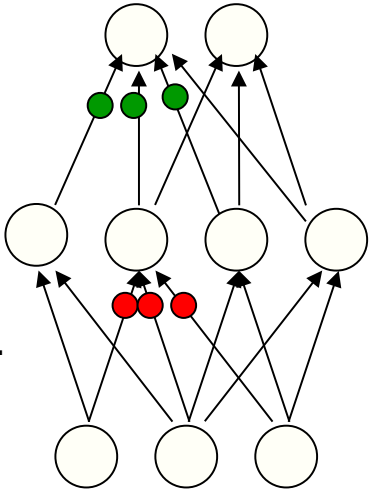
- Networks without hidden units are very limited in the input-output mappings they can model.
- More layers of linear units do not help → Its still linear.
- Fixed output non-linearities are not enough
- We need multiple layers of adaptive non-linear hidden units.
  → This leads to an universal approximator.
- How to train such nets?
- An efficient way of adapting all the weights is needed, not just the last layer.
- Learning the weights going into hidden units is equivalent to learning features.
- Nobody is telling us directly what hidden units should do.

Deep Learning. Vladimir Golkov. Technical University of Munich. Computer Vision Group

Deep Learning – Back Propagation

Learning by perturbing weights

- Randomly perturb one weight and see if it improves performance.
- If so, save the change.
- Very inefficient.
- We need to do multiple forward passes on a representative set of training data just to change one weight.
- Towards the end of learning, large weight perturbations will nearly always make things worse.
- We could randomly perturb all the weights in parallel and correlate the performance gain with the weight changes.
- Not any better because we need lots of trials to "see" the effect of changing one weight through the noise created by all the others.

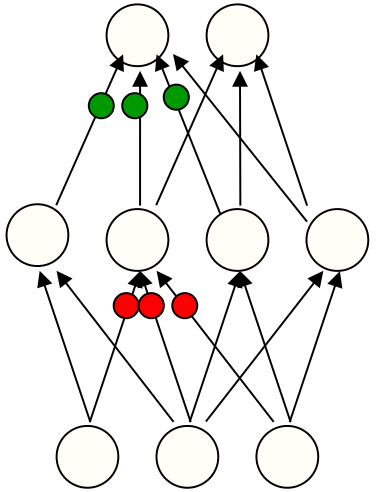


Learning the hidden to output weights is easy.
Learning the input to hidden weights is hard.

Deep Learning – Back Propagation

The idea behind back propagation

- We don't know what the hidden units ought to do, but we can compute how fast the error changes as we change a hidden activity.
- Instead of using desired activities to train the hidden units, use error derivatives w.r.t. hidden activities.
- Each hidden activity can affect many output units and can therefore have many separate effects on the error. These effects must be combined.
- We can compute error derivatives for all the hidden units efficiently.
- Once we have the error derivatives for the hidden activities, its easy to get the error derivatives for the weights going into a hidden unit.



Deep Learning – Back Propagation

Sketch of the back propagation algorithm on a single training case:

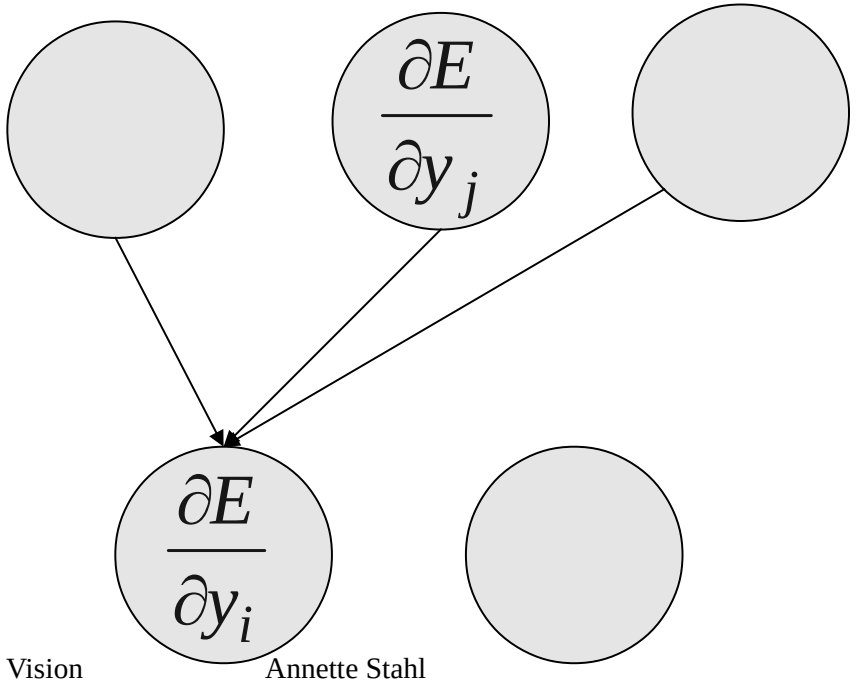
- First convert the discrepancy between each output and its target value into an error derivative.

$$E = \sum_j \frac{1}{2} (y_j - d_j)^2$$

$$\frac{\partial E}{\partial y_j} = y_j - d_j$$

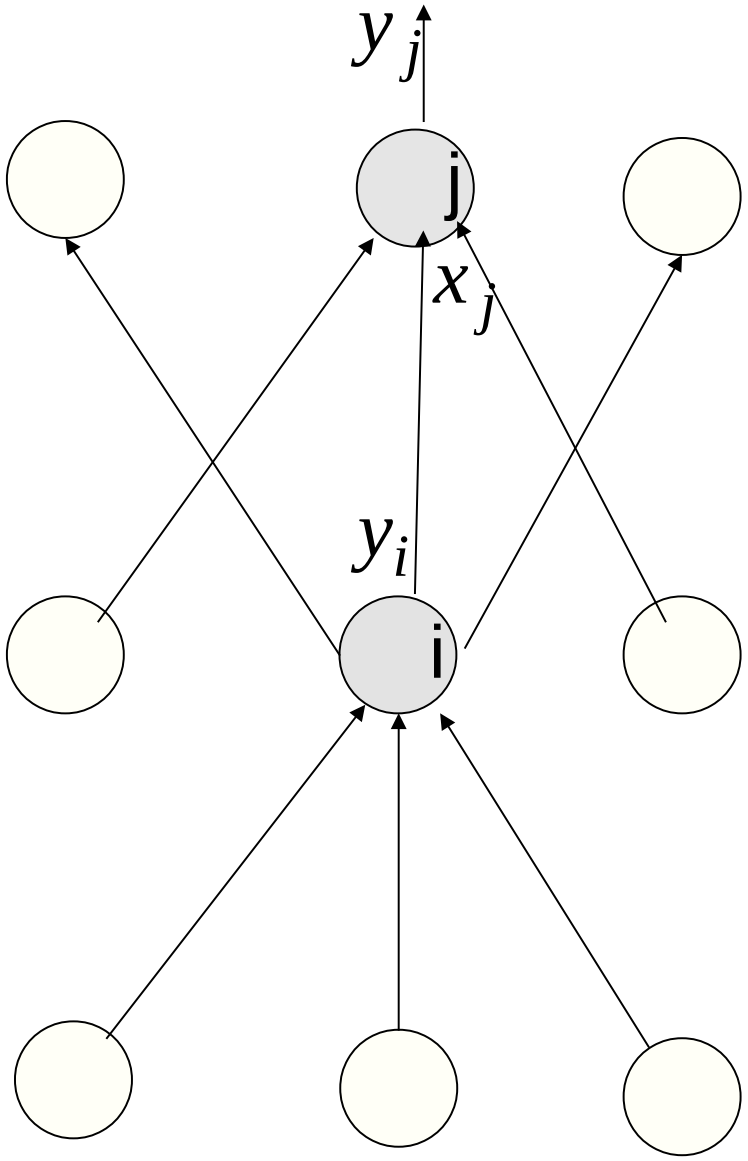
- Then compute error derivatives in each hidden layer from error derivatives in the layer above.

- Then use error derivatives w.r.t. activities to get error derivatives w.r.t. the weights.



Deep Learning – Back Propagation

The derivatives:



$$\frac{\partial E}{\partial x_j} = \frac{dy_j}{dx_j}\frac{\partial E}{\partial y_j} = y_j(1-y_j)\frac{\partial E}{\partial y_j}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial x_j}{\partial w_{ij}}\frac{\partial E}{\partial x_j} = y_i\frac{\partial E}{\partial x_j}$$

$$\frac{\partial E}{\partial y_i} = \sum_j \frac{dx_j}{dy_i}\frac{\partial E}{\partial x_j} = \sum_j w_{ij}\frac{\partial E}{\partial x_j}$$

Deep Learning – Back Propagation

Back-propagation has been used for a large number of practical applications.

- Recognizing hand-written characters
- Predicting the future price of stocks
- Detecting credit card fraud
- Recognize speech
- Predicting the next word in a sentence from the previous words

  → This is essential for good speech recognition.

Deep Learning

2D Convolutional Layer

Appropriate for 2D structured data (e.g. images) where we want:

- Locality of feature extraction (far-away pixels do not influence local output)
- Translation-equivariance (shifting input in space (i, j dimensions) yields same output shifted in the same way)

$$x_{i,j,k}^{(l)} = s_l\left(b_k^{(l)} + \sum_{\hat{\imath},\hat{\jmath},\hat{k}} w_{i-\hat{\imath},j-\hat{\jmath},\hat{k},k}^{(l)} x_{i,j,\hat{k}}^{(l-1)}\right)$$

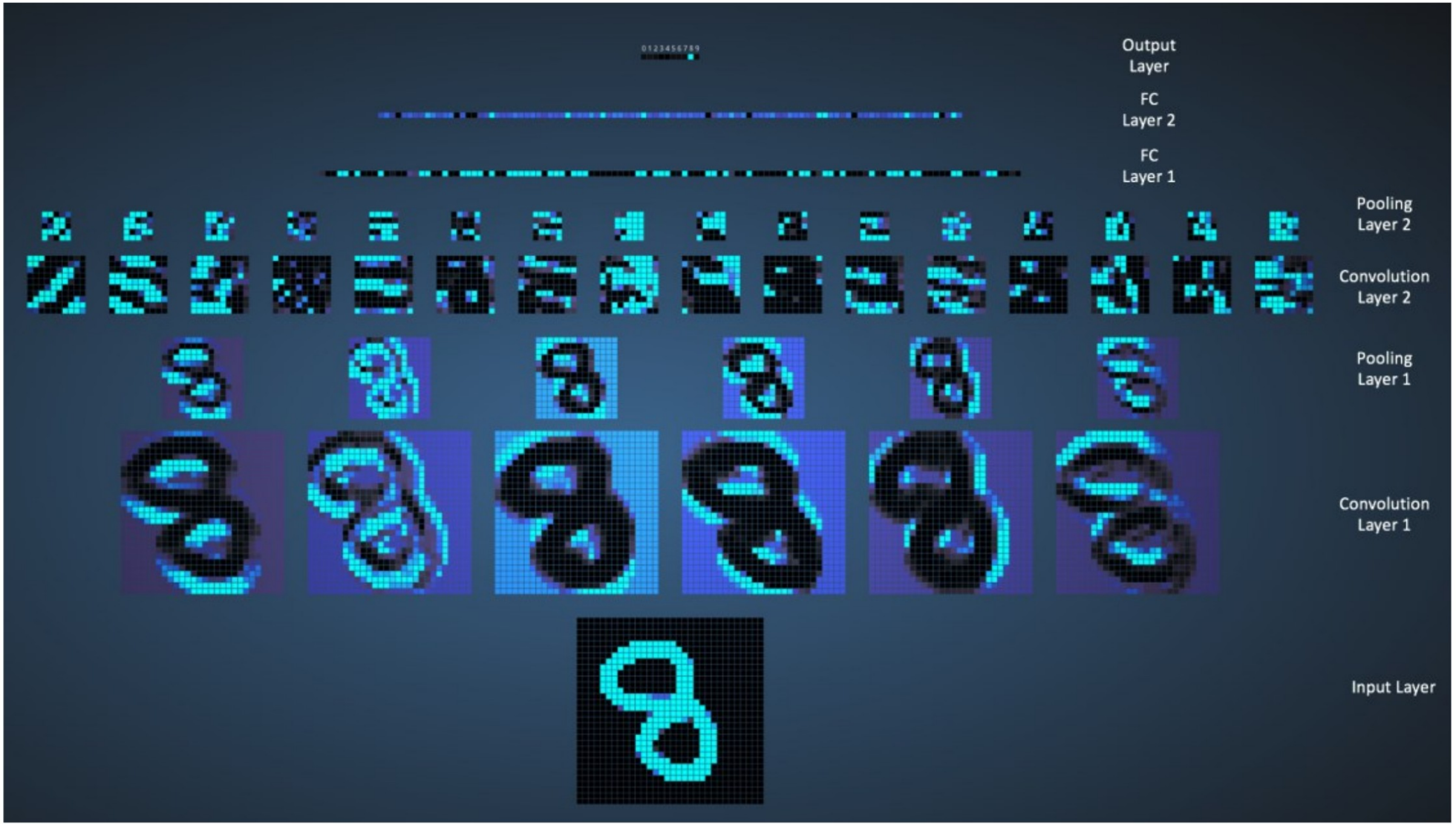
- the size of W along the i, j dimensions is called “filter size”
- the size of W along the $\hat{k}$ dimension is the number of input channels (e.g. three (red, green, blue) in first layer)
- the size of W along the k dimension is the number of filters (number of output channels)

- Equivalent for 1D, 3D, ...

http://cs231n.github.io/assets/conv-demo/

Deep Learning. Vladimir Golkov. Technical University of Munich. Computer Vision Group

Deep Learning



http://scs.ryerson.ca/~aharley/vis/conv/flat.html

Deep Learning. Vladimir Golkov. Technical University of Munich. Computer Vision Group

Deep Learning

Loss Functions

N-class classification:
- N outputs
- nonlinearity in last layer: softmax
- loss: categorical cross-entropy between outputs $x^{(L)}$ and targets t (sum over all training samples)

2-class classification:
- 1 output
- nonlinearity in last layer: sigmoid
- loss: binary cross-entropy between outputs $x^{(L)}$ and targets t (sum over all training samples)

2-class classification (alternative formulation)
- 2 outputs
- nonlinearity in last layer: softmax
- loss: categorical cross-entropy between outputs $x^{(L)}$ and targets t (sum over all training samples)

Many regression tasks:
- linear output in last layer
- loss: mean squared error between outputs $x^{(L)}$ and targets t (sum over all training samples)

Deep Learning. Vladimir Golkov. Technical University of Munich. Computer Vision Group

Deep Learning

Neural Network Training Procedure

- Fix number L of layers
- Fix sizes of weight arrays and bias vectors
  - For a fully-connected layer, this corresponds to the “number of neurons”
- Fix nonlinearities
- Initialize weights and biases with random numbers
- Repeat:
  - Select mini-batch (i.e. small subset) of training samples
  - Compute the gradient of the loss with respect to all trainable parameters (all weights and biases)
    - Use chain rule (“error backpropagation”) to compute gradient for hidden layers
  - Perform a gradient-descent step (or similar) towards minimizing the error
    - (Called “stochastic” gradient descent because every mini-batch is a random subset of the entire training set)
    - Important hyperparameter: learning rate (i.e. step length factor)
  - “Early stopping”: Stop when loss on validation set starts increasing (to avoid overfitting)

Deep Learning. Vladimir Golkov. Technical University of Munich. Computer Vision Group

Deep Learning

Data Representation

- The data representation should be natural
  (do not “outsource” known data transformations to the learning of the mapping)

- Make it easy for the network
  - For angles, we use sine and cosine to avoid the jump from 360° to 0°
    - Redundancy is okay!
  - Fair scale of features (and initial weights and learning rate) to facilitate optimization

- Data augmentation using natural assumptions

- Features from different distributions or missing: use several disentangled inputs to tell the network!
  - Trade-off: The more a network should be able to do, the *much* more data and/or better techniques are required

- https://en.wikipedia.org/wiki/Statistical_data_type
  - Categorical variable: one-hot encoding
  - Ordinal variable: cumulative sum of one-hot encoding [cf. Jianlin Cheng, arXiv:0704.1028]
  - etc

Deep Learning. Vladimir Golkov. Technical University of Munich. Computer Vision Group

Deep Learning

Regularization to Avoid Overfitting

Impose meaningful invariances/assumptions about mapping in a hard or soft manner:

- Limited complexity of model: few layers, few neurons, weight sharing
- Locality and shift-equivariance of feature extraction: ConvNets
- Exact spatial locations of features don't matter: pooling; strided convolutions
  - http://cs231n.github.io/assets/conv-demo/
- Deep features shouldn't strongly rely on each other: dropout (randomly setting some deep features to zero during training)
- Data augmentation:
  - Known meaningful transformations of training samples
  - Random noise (e.g. dropout) in first or hidden layers
- Optimization algorithm tricks, e.g. early stopping

Deep Learning. Vladimir Golkov. Technical University of Munich. Computer Vision Group

Deep Learning for Object Detection

Based on the whether following the “proposal and refine”

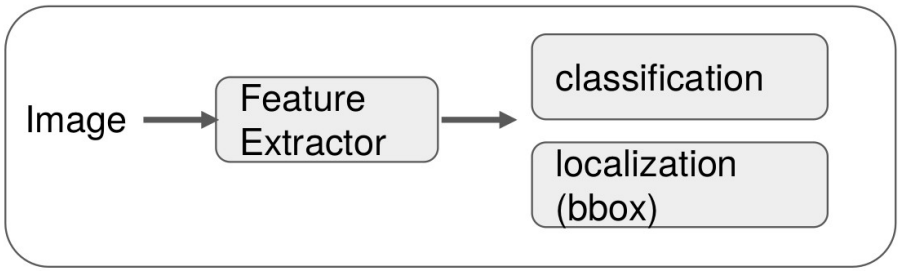
One Stage

- Example: Densebox, YOLO (YOLO v2), SSD, Retina Net
- Keyword: Anchor, Divide and conquer, loss sampling

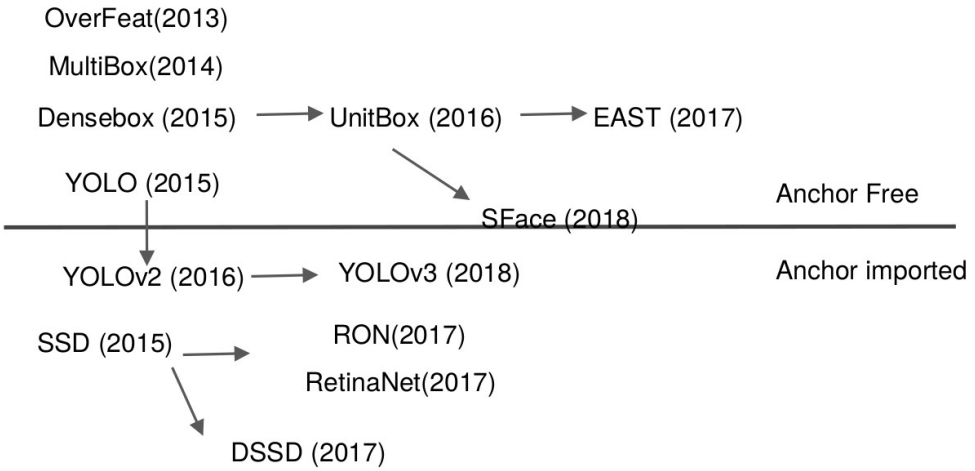
Two Stage

- Example: RCNN (Fast RCNN, Faster RCNN), RFCN, FPN, MaskRCNN
- Keyword: speed, performance

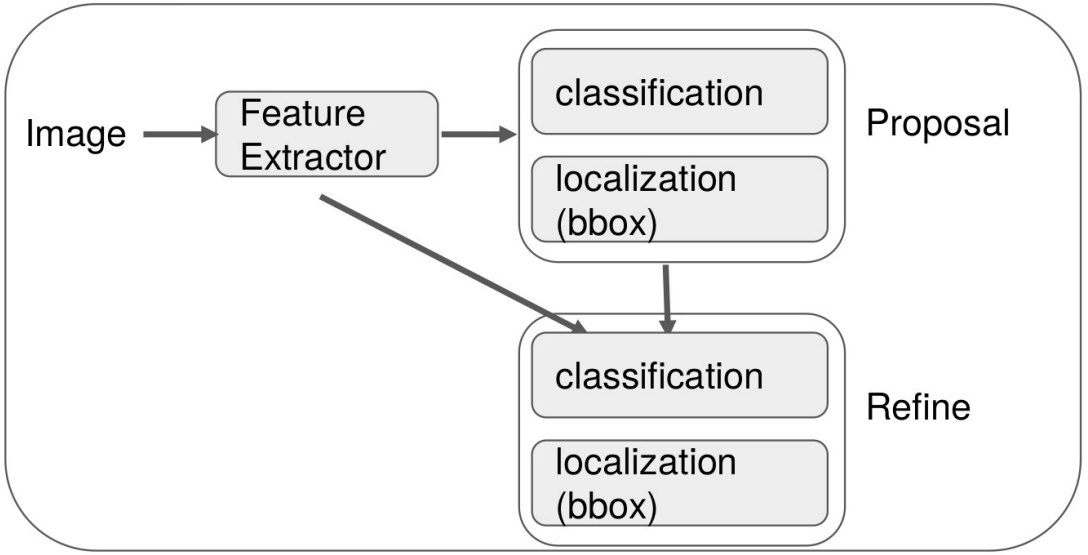
Deep Learning – Brief History

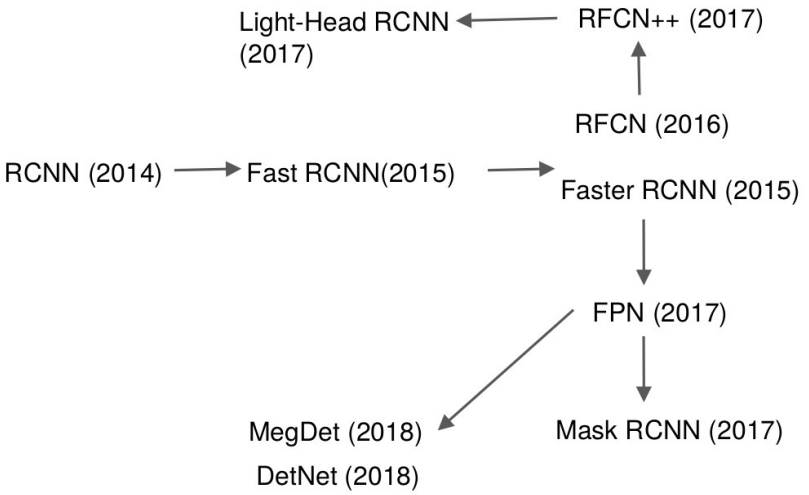


One stage detector



two stages detector





Deep Learning

One stage Detector: You Only Look Once - YOLO (Redmon et al. 2016)

- Detection as Single Regression Problem
- Developed as Single Convolutional Network
- Reason Globally on the Entire Image
- Learns Generalizable Representations Easy & Fast
- Can process 45 frames per second on a GPU (155 fps for Fast YOLO)
- Lower mAP than some R-CNN variants, but much faster
- Highest mAP of real-time detectors (≥30fps)
-
-

Limitations:

- Struggle with Small Object.
- Loss function threats errors in different boxes ratio at the same.
- Struggle with Different aspects and ratios of objects.
- Loss function is an approximation.

Deep Learning

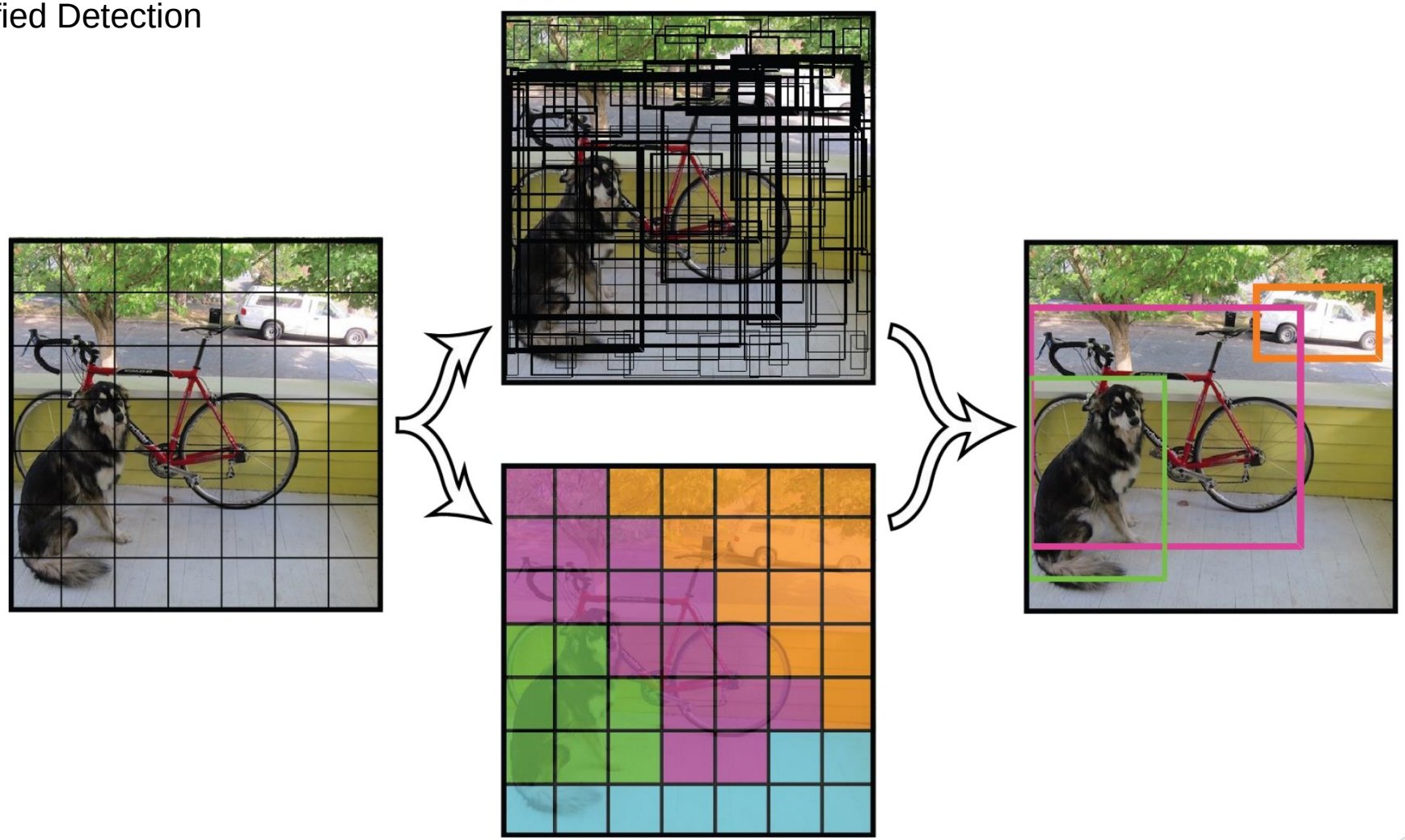
You Only Look Once - YOLO (Redmon et al. 2016)

- Divides image into a $S \times S$ grid
  - If the center of an object fall into a grid cell, it will be the responsible for the object.
- Each grid cell predicts
  - B bounding boxes, each as (x, y, w, h) (coordinates,width, height) and
  - a confidence (five total predictions) $x, y, w, h \in [0, 1]$ (relative to image dimensions and grid cell location)
  - C class probabilities
- Output is a $S \times S \times (5B + C)$ tensor

Deep Learning

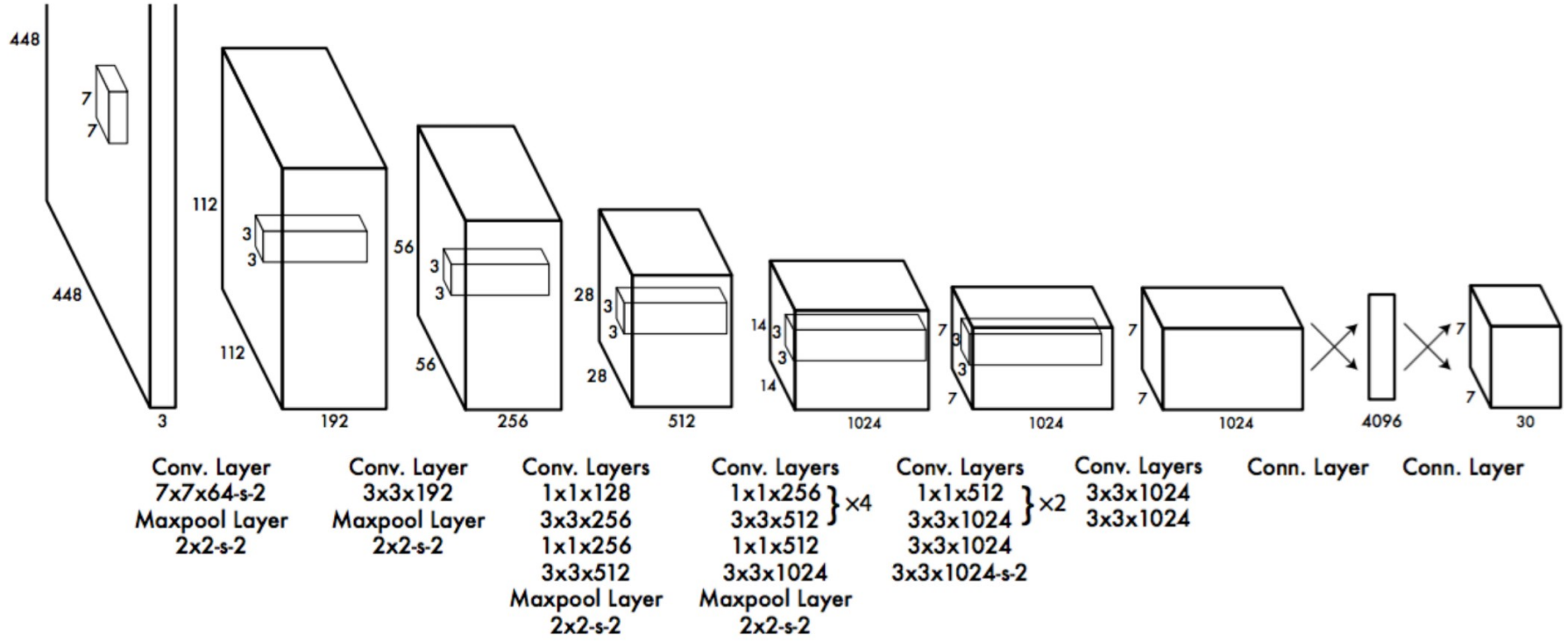
You Only Look Once - YOLO (Redmon et al. 2016)

Unified Detection



Deep Learning

You Only Look Once - YOLO (Redmon et al. 2016)



Yolo - Demo

https://pjreddie.com/darknet/yolo/

https://youtu.be/NM6lrxy0bxs?list=PLrrmP4uhN47Y-hWs7DVfCmLwUACRigYyT

Mask R-CNN for Object Detection and Segmentation

K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, 2017, pp. 2980-2988.

https://github.com/matterport/Mask_RCNN

https://arxiv.org/abs/1703.06870

Literature

- Szeliski Ch. 14.1.
- Szeliski Ch. 14.2.
- Szeliski Ch. 14.4.1
- The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman

  https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

The SVM classifier. C19 Machine Learning. Hilary 2015 A. Zisserman
https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf