



LUND
UNIVERSITY

350

Image Analysis (FMAN20)

Lecture 2, 2018

MAGNUS OSKARSSON

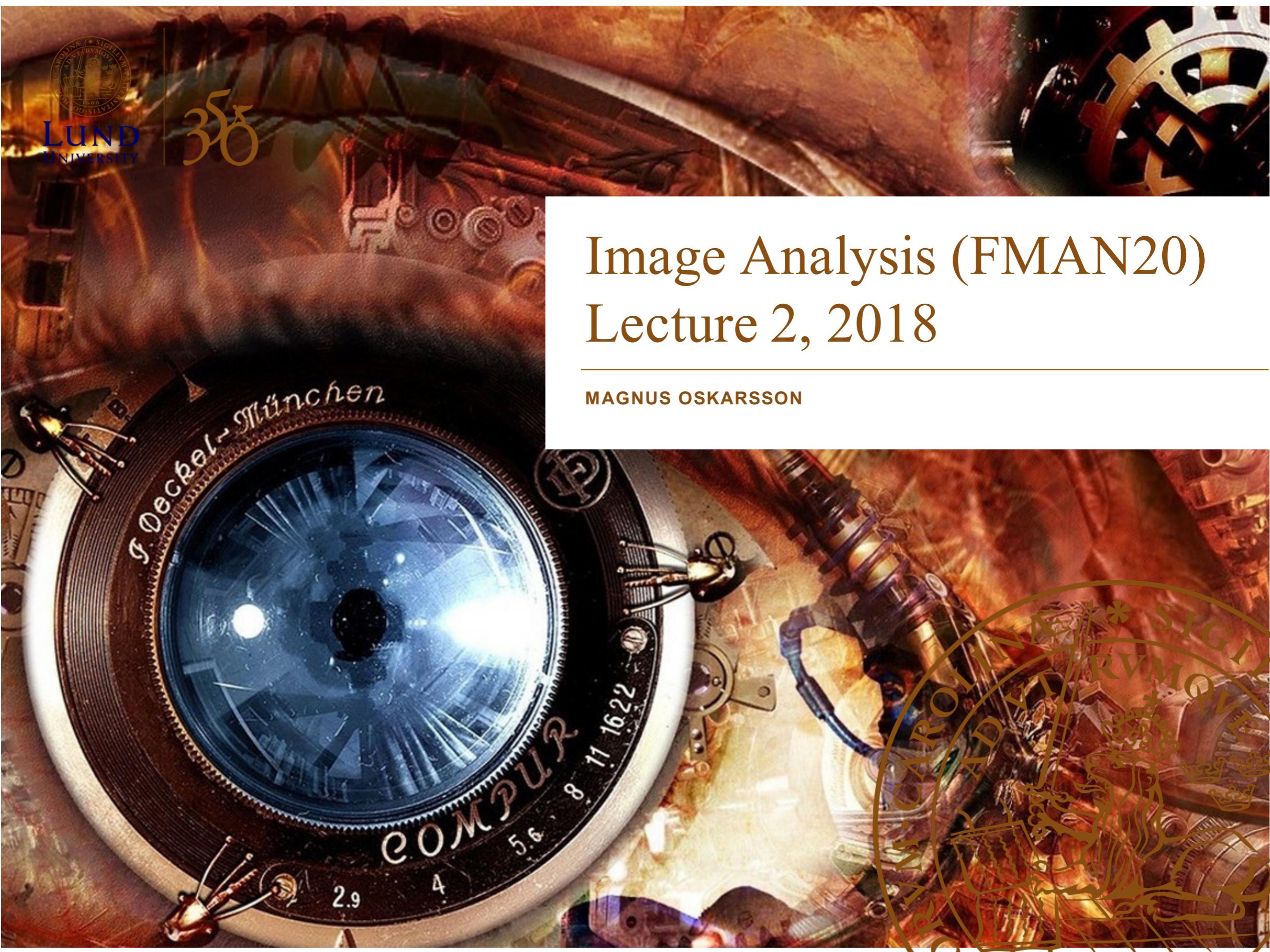


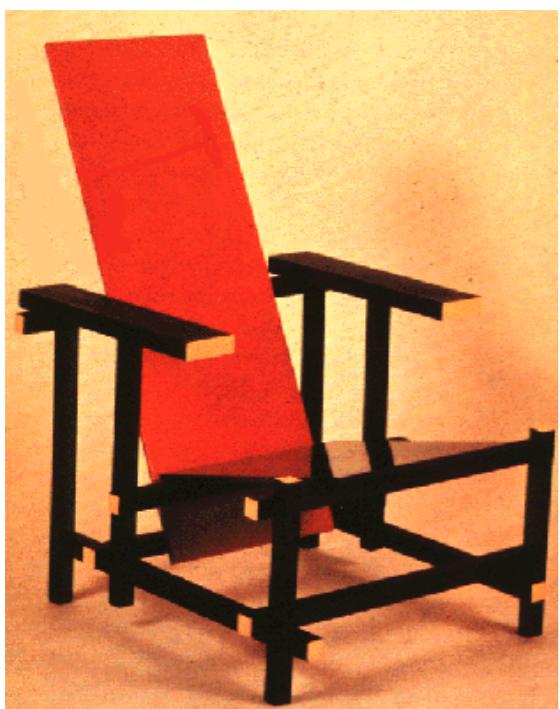
Image Analysis - Motivation

Overview – Machine Learning 1

1. Machine Learning
2. Bayes Theorem
 1. Counting
 2. Binning
 3. Curse of dimensionality
 4. Adaptive binning (K-means)
3. Nearest Neighbour, K-NN
4. Parametric Density Estimation (Plug-in classifier)
5. Logistic regression

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Challenges: object intra-class variation



•slide credit: Fei-Fei, Fergus & Torralba

In this course

- Tools:
 - Basics of image analysis
 - Linear Algebra, Linear System Theory
 - Mathematical Statistics
 - **Machine Learning**
 - Segmentation
- System development
 - Based on the tools
 - Ground truth, evaluation, benchmarking

Object recognition (in mobile phones)



- This is becoming real:
 - **Lincoln** Microsoft Research
 - [Point & Find, Nokia](#)
 - [SnapTell.com \(now amazon\)](#)

Optical character recognition (OCR)

- Technology to convert scanned docs to text
 - If you have a scanner, it probably came with OCR software



- Digit recognition, AT&T labs
- <http://www.research.att.com/~yann/>

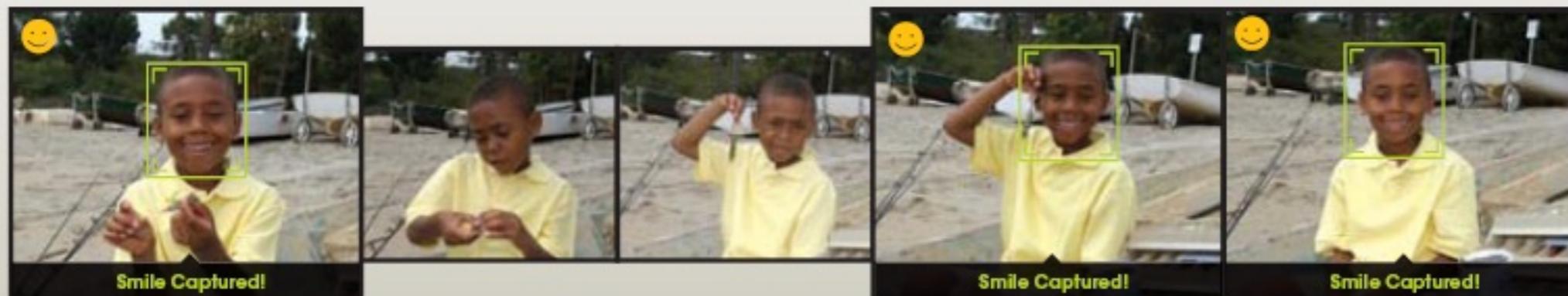
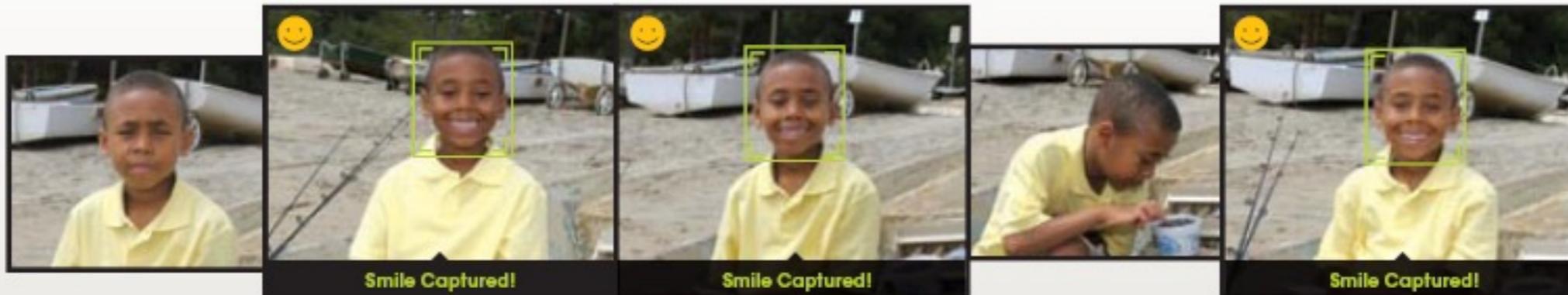


- License plate readers
- http://en.wikipedia.org/wiki/Automatic_number_plate_recognition

Smile detection?

The Smile Shutter flow

Imagine a camera smart enough to catch every smile! In Smile Shutter Mode, your Cyber-shot® camera can automatically trip the shutter at just the right instant to catch the perfect expression.



- Sony Cyber-shot® T70 Digital Still Camera

Machine Learning – classify

All of these classification problems have in common:

- ▶ data - \mathbf{x} (after segmentation, extract features)
- ▶ A number of classes

One would like to determine a class for every possible feature vector.

Here we will assume that the features are represented as a column vector, i.e. $\mathbf{x} \in \mathbb{R}^n$,

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

One would like to compare the feature vector \mathbf{x} with those that one usually gets with a number of classes. Let y denote the class index, i.e. the classes are $y \in \omega_y = \{1, \dots, M\}$ where M denotes the number of classes.

Typical system: Image - filtering - segmentation - features - classification

Machine Learning – Bayes rule

Assume that one feature vector \mathbf{x} and class y are drawn from a joint probability distribution. If one can calculate the probability that the class is $y = j$ given the measurements \mathbf{x} , i.e. the so called **posterior probability**.

$$P(y = j|\mathbf{x})$$

The **maximum a posteriori classifier** is obtained as selecting the class j that maximizes the posterior probability, i.e.

$$j = \operatorname{argmax}_k P(y = k|\mathbf{x}).$$

It is often easier to model and estimate the **likelihood** $P(\mathbf{x}|y = j)$ and to model the **prior** $p(y = j)$. The a posteriori probabilities can then be calculated using the Bayes rule,

$$p(y = j|\mathbf{x}) = \frac{p(\mathbf{x}|y = j)p(y = j)}{p(\mathbf{x})}.$$

Example

- In a small town, there are two bicycle brands **albatross** and **butterfly**. Albatross sell mostly **green** bikes (80 percent) are green and the rest are **yellow**. Butterfly sell 50 percent green bikes and 50 percent yellow. The Albatross brand is more popular. They have 90 percent of the market in the town. **In another nearby town**, from a distance you see a yellow bike. What is the probability that the bike is an Butterfly bike?

Joint probabilities $P(x,y)$

- Joint probability and the prior $P(x) = \sum_y P(x,y)$
- Joint probability and the total probability $P(y) = \sum_x P(x,y)$

$$P(y) = \sum_x P(x,y)$$

$P(x,y)$	x=Green	x=Yellow
y=Albatross	0,72	0,18
y=Butterfly	0,05	0,05
	0,77	0,23
p(x=Green) p(x=Yellow)		

0,9 $p(y=Albatross)$
0,1 $p(y=Butterfly)$

1

$P(x y)$	x=Green	x=Yellow	
y=Albatross	0,80	0,20	1,00
y=Butterfly	0,50	0,50	1,00
	1,30	0,70	2,00

$$P(x) = \sum_y P(x,y)$$

A posteriori probabilités

- A posteriori probabilités

$$P(\mathbf{y}|\mathbf{x}) = P(\mathbf{x}, \mathbf{y})/P(\mathbf{x})$$

P(x,y)	x=Green	x=Yellow	
y=Albatross	0,72	0,18	0,9 p(y=Albatross)
y=Butterfly	0,05	0,05	0,1 p(y=Butterfly)
	0,77	0,23	1
p(x=Green)		p(x=Yellow)	

P(x y)	x=Green	x=Yellow	
y=Albatross	0,80	0,20	1,00
y=Butterfly	0,50	0,50	1,00
	1,30	0,70	2,00

P(y x)	x=Green	x=Yellow	
y=Albatross	0,94	0,78	1,72
y=Butterfly	0,06	0,22	0,28
1	1	2	

Usual workflow

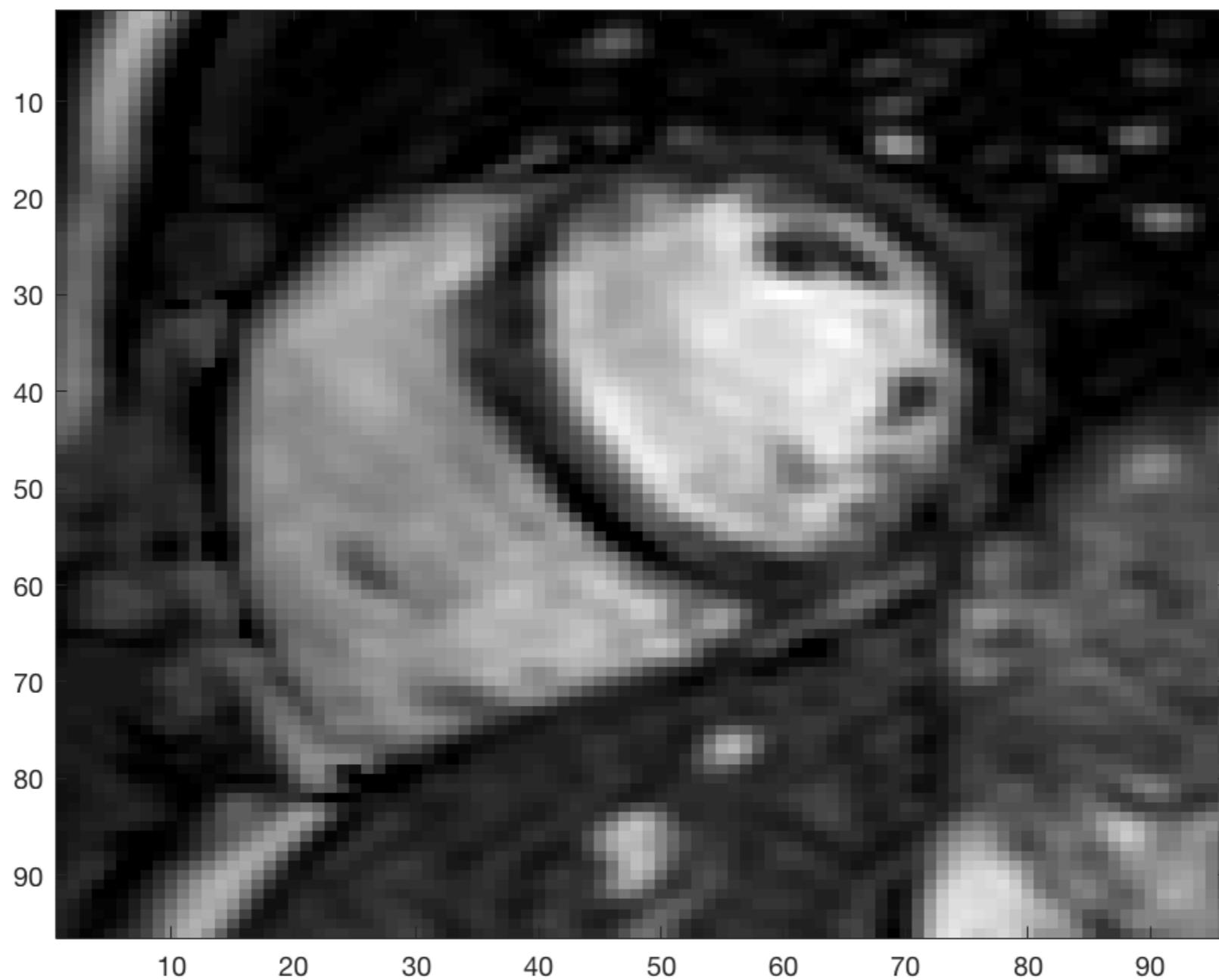
- Model prior $P(y)$ and measurement probabilities $P(x|y)$
- Joint probability $P(x,y) = P(x|y)P(y)$
- Total probability $P(x) = \sum_y P(x,y)$
- A posteriori probability $P(y|x) = P(x,y)/P(x)$
- Classify according to maximum a posteriori probability

$P(x,y)$	x=Green	x=Yellow	
y=Albatross	0,72	0,18	0,9 p(y=Albatross)
y=Butterfly	0,05	0,05	0,1 p(y=Butterfly)
	0,77	0,23	1
p(x=Green)		p(x=Yellow)	

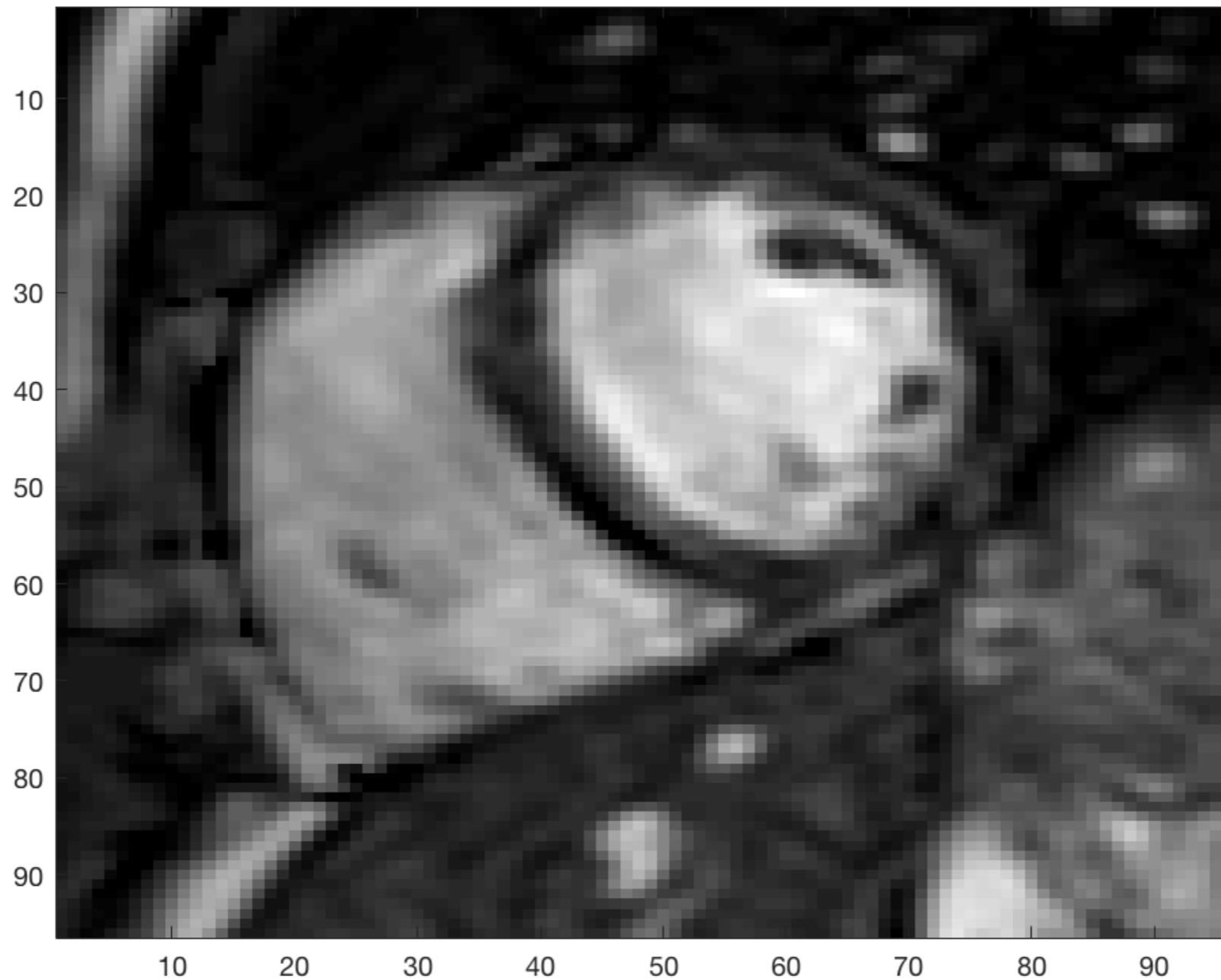
$P(x y)$	x=Green	x=Yellow	
y=Albatross	0,80	0,20	1,00
y=Butterfly	0,50	0,50	1,00
	1,30	0,70	2,00

$P(y x)$	x=Green	x=Yellow	
y=Albatross	0,94	0,78	1,72
y=Butterfly	0,06	0,22	0,28
1	1	2	

Another example – heart pixels



Use binning (quantization) to get fewer gray levels. Here 6 levels



Discretize pixel brightness
using 6 bins. Estimate probabilities
from training data

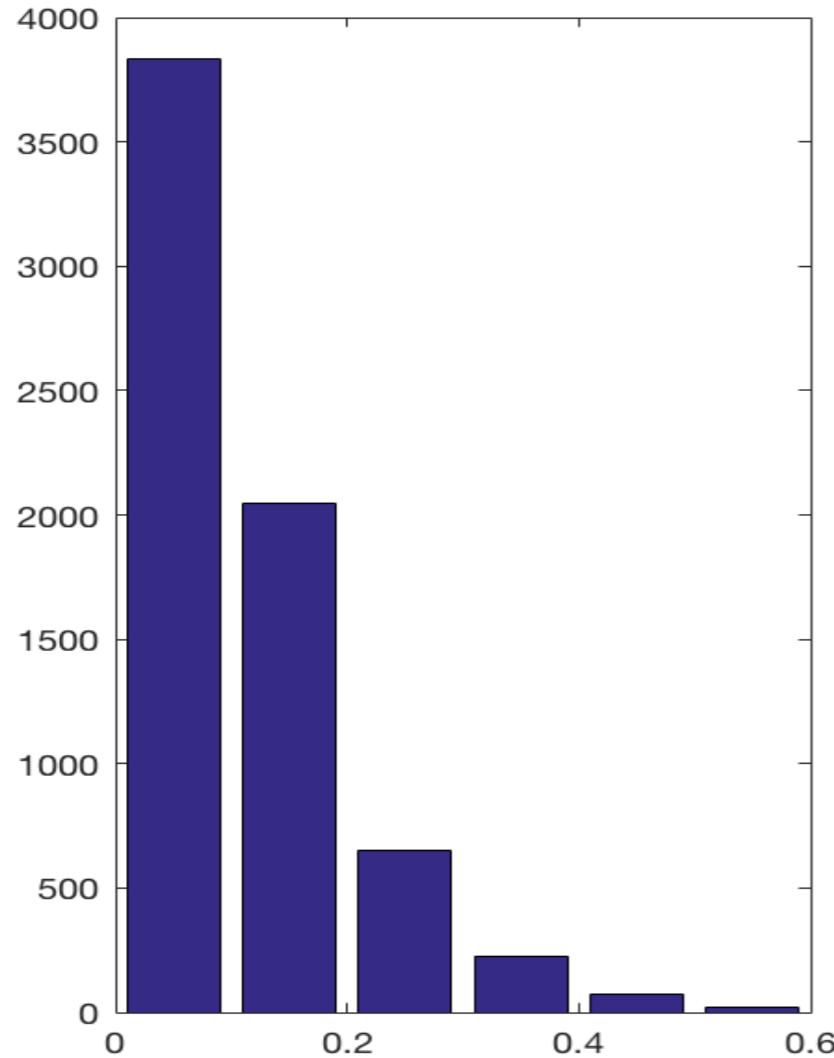
$P(x | \text{background})$

$P(x | \text{heart})$

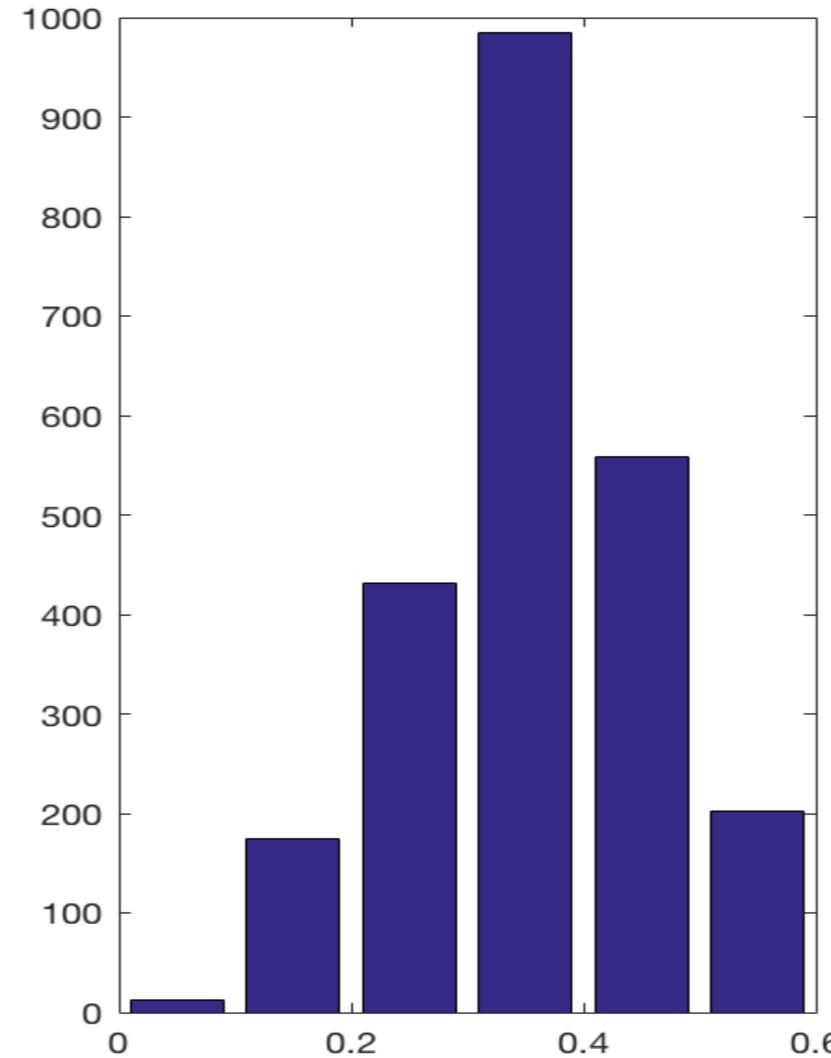
$P(x y)$	x=1	x=2	x=3	x=4	x=5	x=6	
y=heart	0,0051	0,0737	0,1825	0,417	0,2363	0,0855	1,00
y=background	0,56	0,30	0,095	0,03	0,01	0,00	1,00
	0,56	0,37	0,28	0,45	0,25	0,09	2,00

Discretize pixel brightness
using 6 bins. Estimate probabilities
from training data

$P(x | \text{background})$



$P(x | \text{heart})$



Estimate a posteriori probabilités.

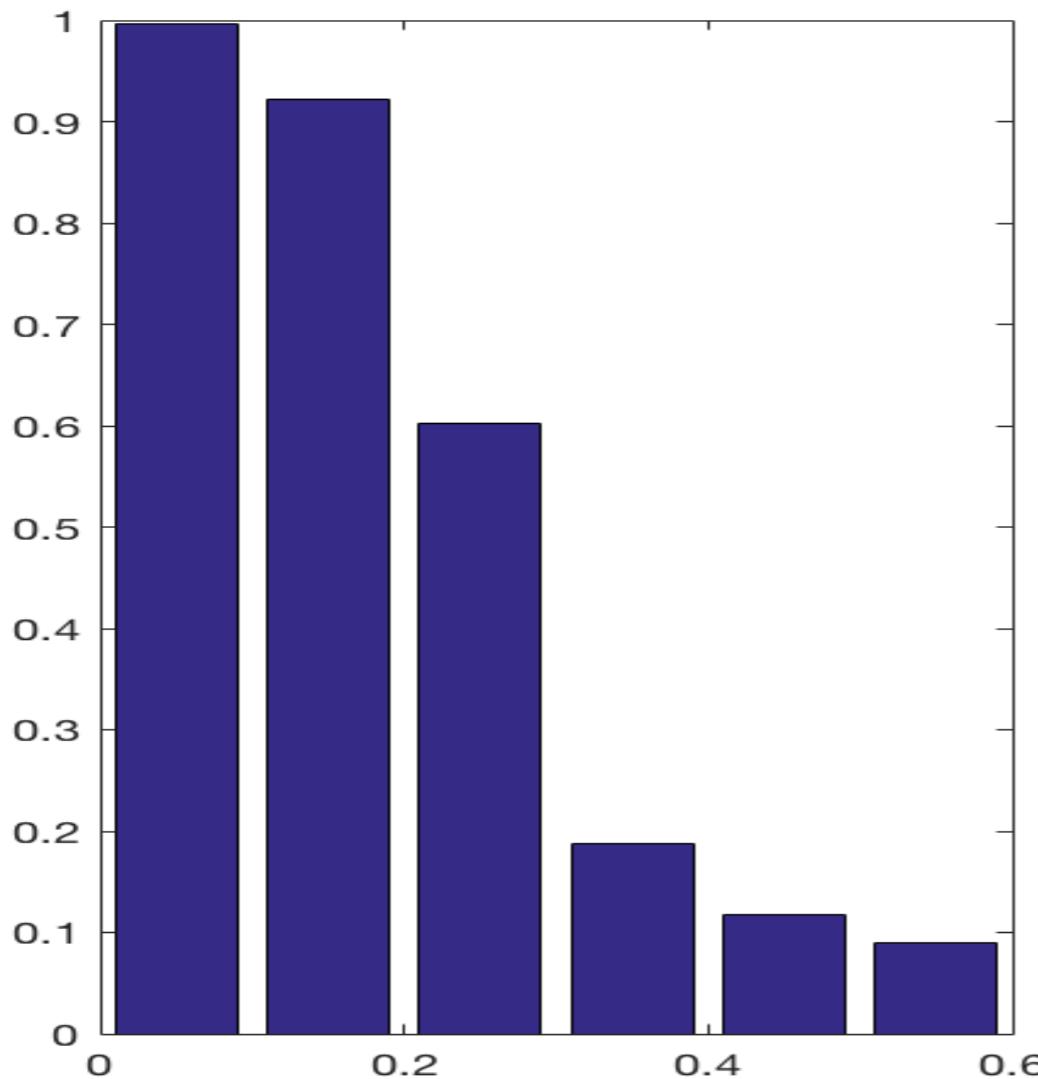
P(x y)	x=1	x=2	x=3	x=4	x=5	x=6	
y=heart	0,0051	0,0737	0,1825	0,417	0,2363	0,0855	1,00
y=background	0,56	0,30	0,095	0,03	0,01	0,00	1,00
	0,56	0,37	0,28	0,45	0,25	0,09	2,00

P(x,y)	x=1	x=2	x=3	x=4	x=5	x=6	
y=heart	0,001326	0,019162	0,04745	0,10842	0,061438	0,02223	0,26 p(y=heart)
y=background	0,413956	0,221112	0,0703	0,024494	0,007992	0,002146	0,74 p(y=background)
	0,415282	0,240274	0,11775	0,132914	0,06943	0,024376	1
	p(x=1)	p(x=2)	p(x=3)	p(x=4)	p(x=5)	p(x=6)	

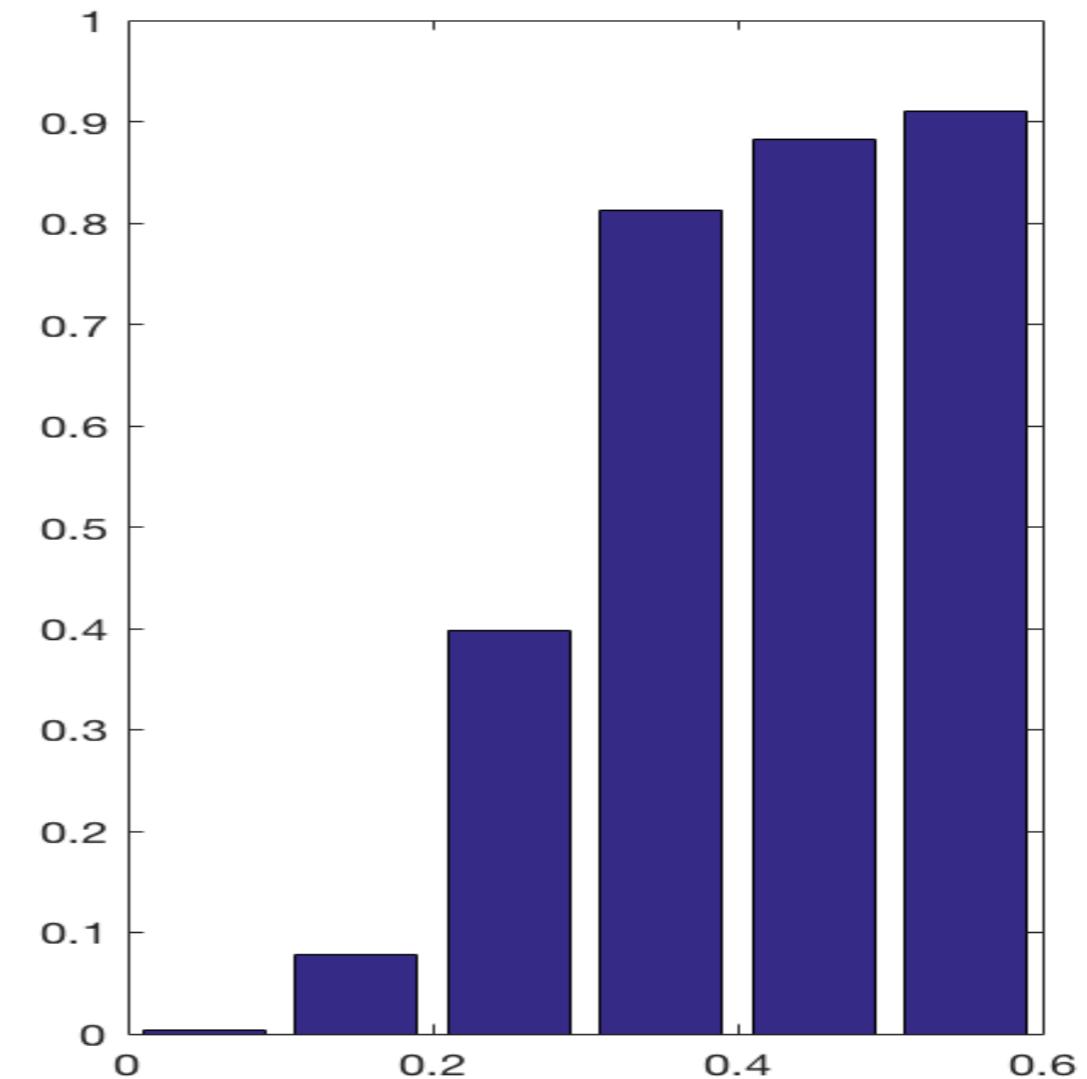
P(y x)	x=1	x=2	x=3	x=4	x=5	x=6	
y=heart	0,00	0,08	0,40	0,82	0,88	0,91	0,92
y=background	1,00	0,92	0,60	0,18	0,12	0,09	1,08
	1	1	1	1	1	1	2

Estimate a posteriori probabilités.
Use these as gray level transforms

$P(\text{background} \mid x)$



$P(\text{heart} \mid x)$



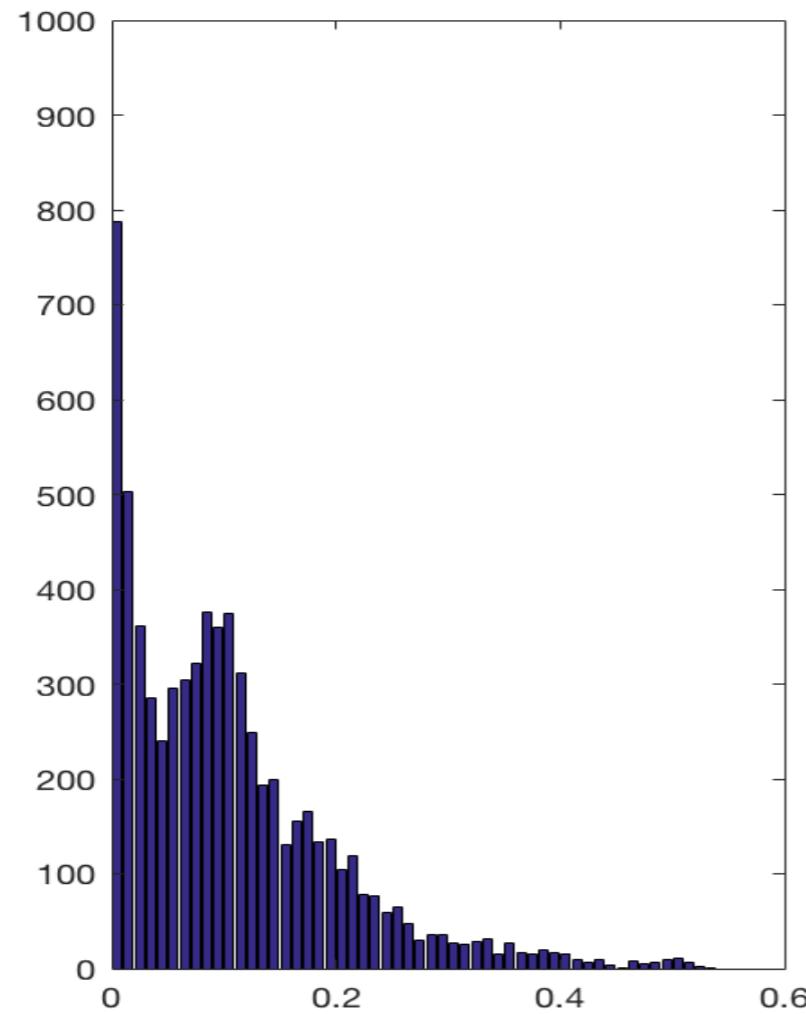
Discretize pixel brightness
using 6 bins. Estimate probabilities

$$P(\text{heart} \mid x)$$

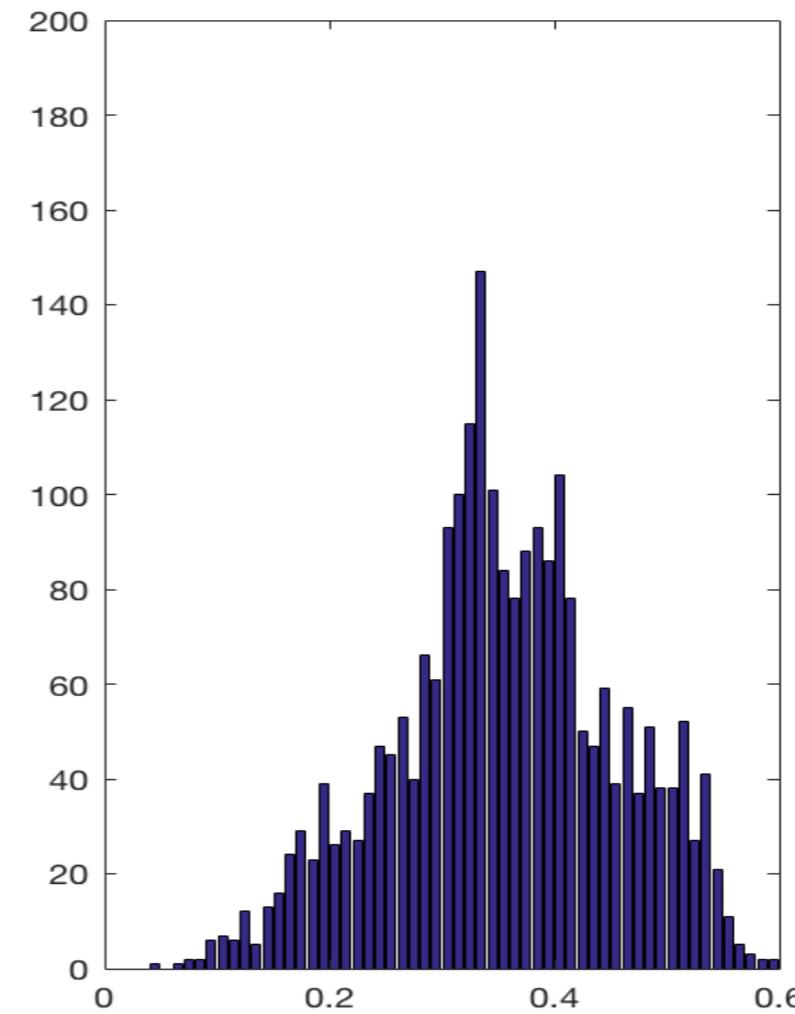


Discretize pixel brightness
using 60 bins. Estimate probabilites

$P(x | \text{background})$

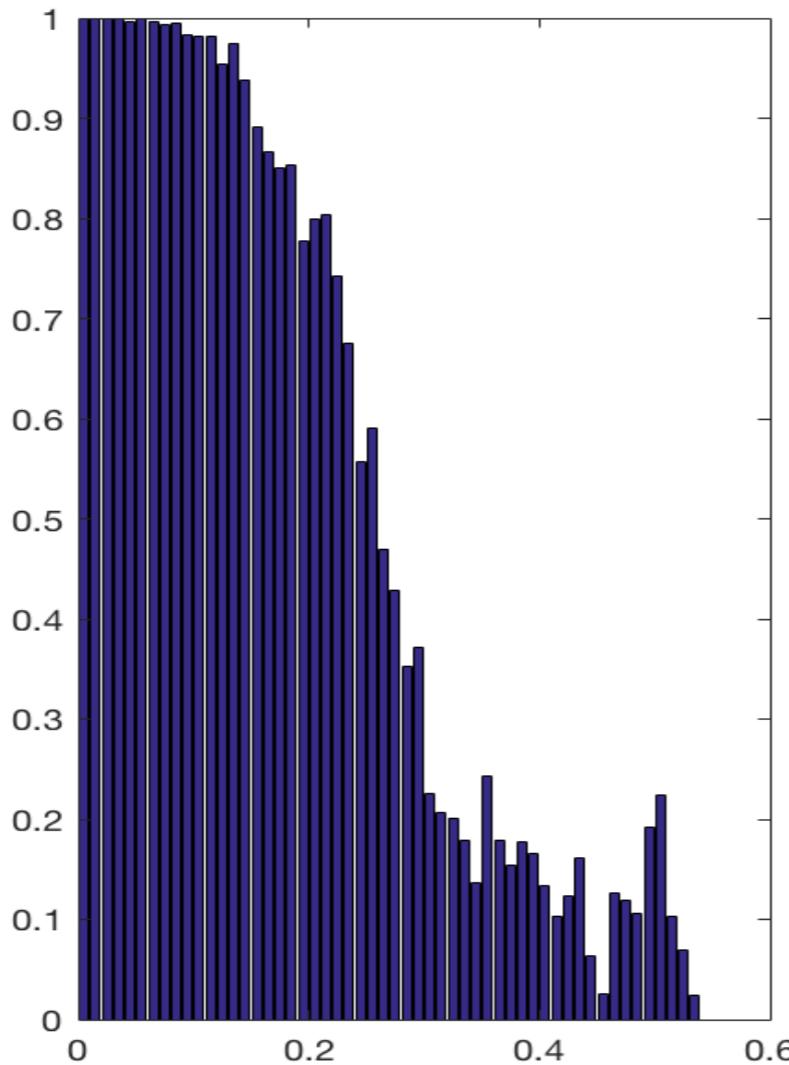


$P(x | \text{heart})$

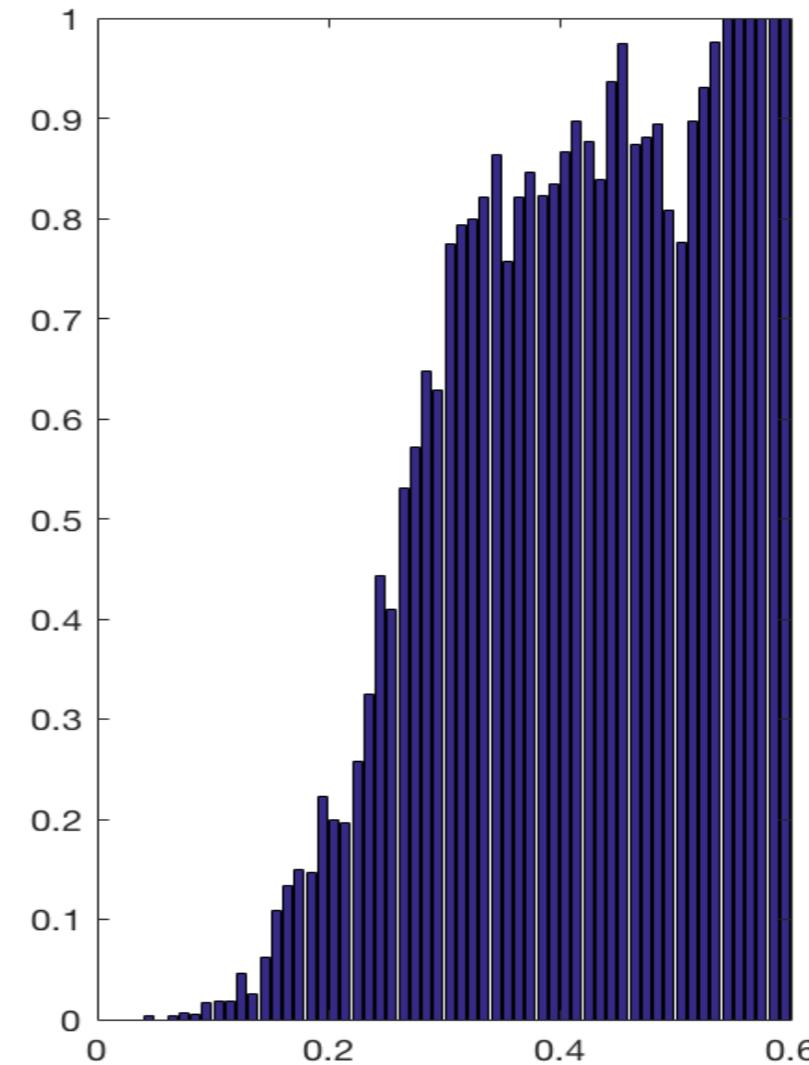


Estimate a posteriori probabilités.
Use these as gray level transforms

$P(\text{background} \mid x)$

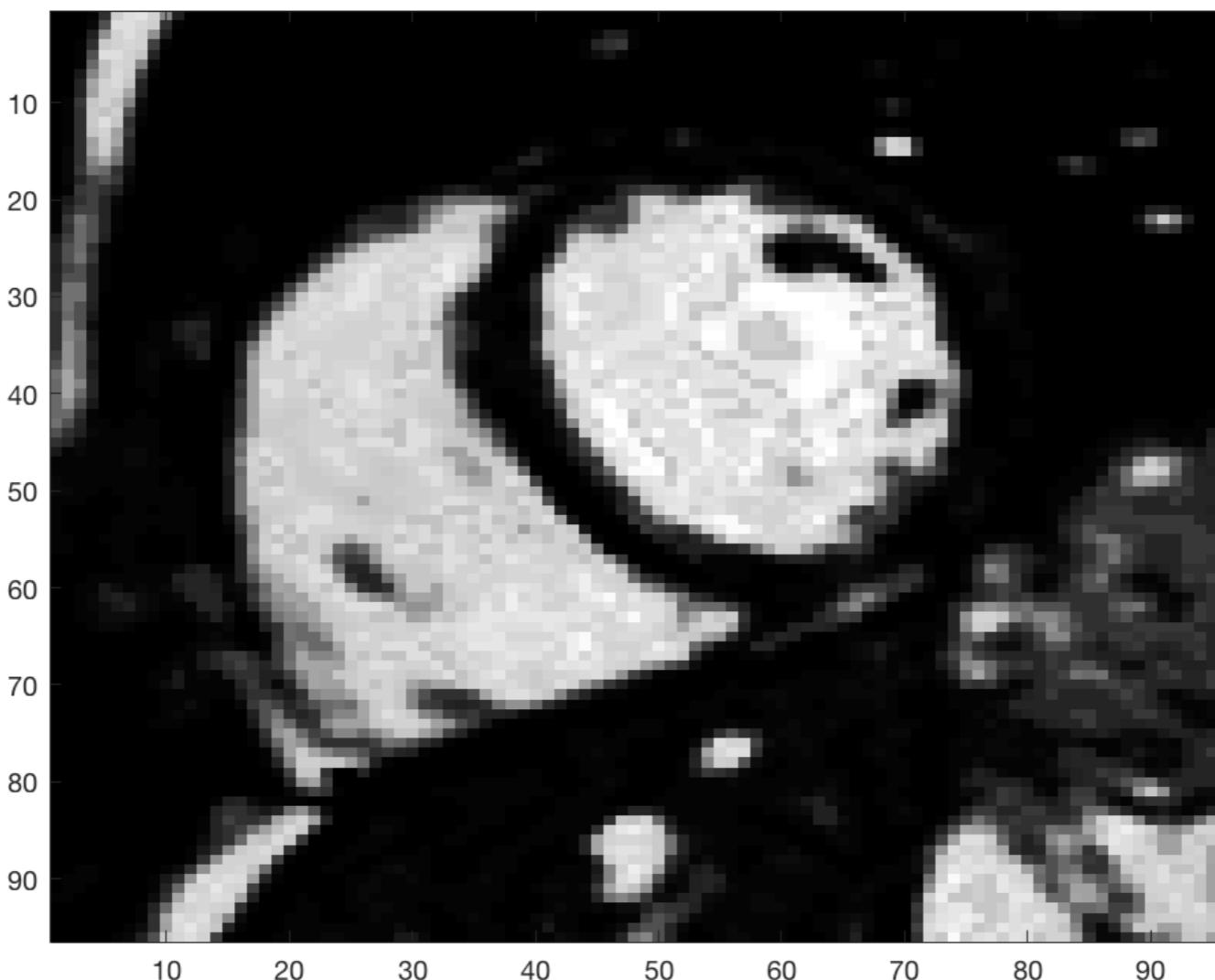


$P(\text{heart} \mid x)$



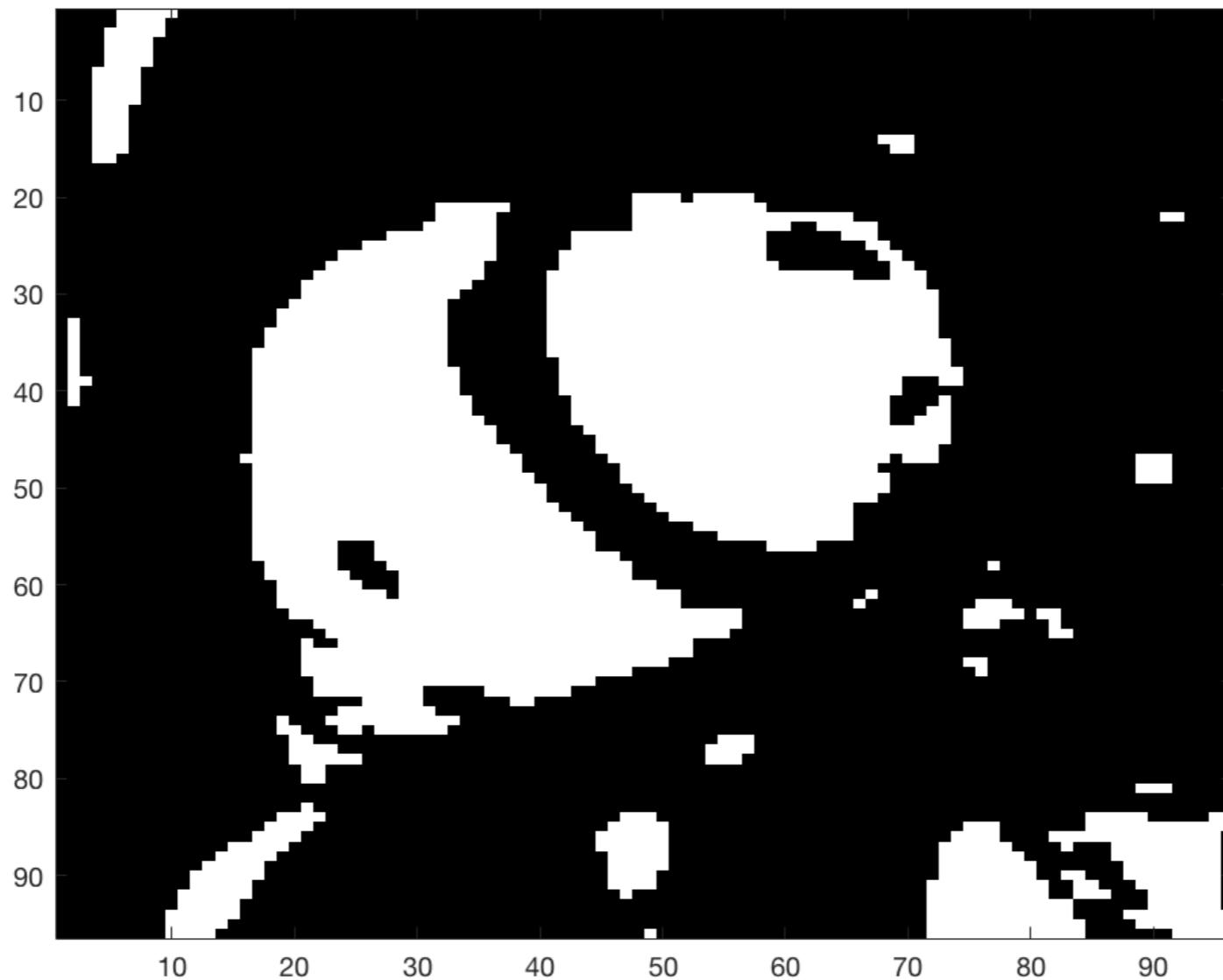
Discretize pixel brightness
using 60 bins. Estimate probabilites

$$P(\text{heart} \mid x)$$



Discretize pixel brightness
using 60 bins. Estimate probabilites

$$P(\text{heart} \mid x) > 0.5$$

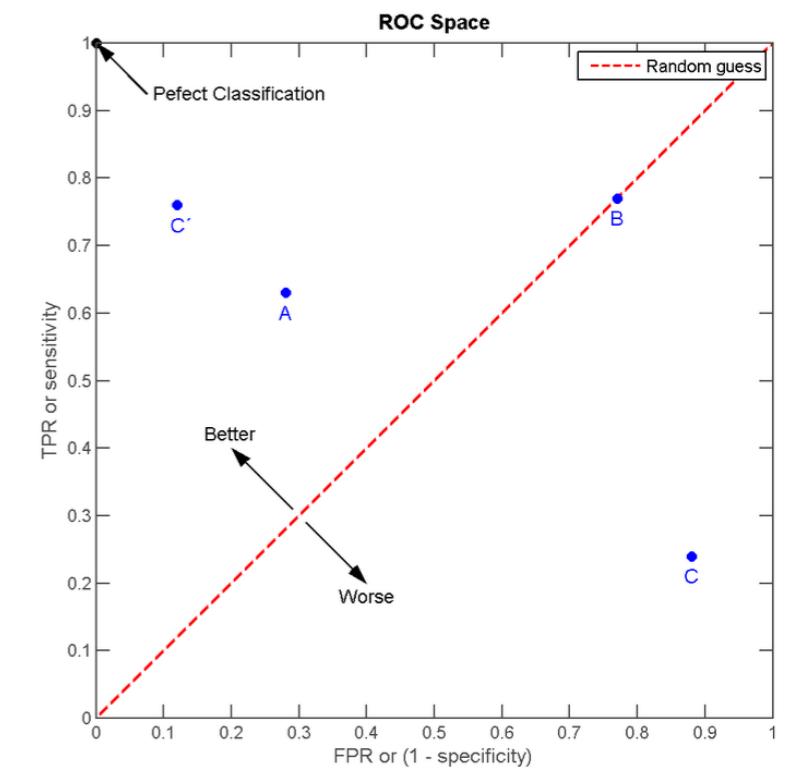
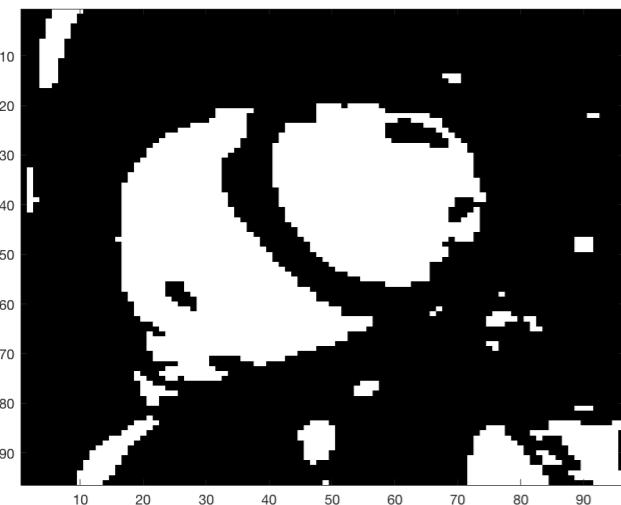


False Positives, False Negatives

ROC - Curve

- For two class problems - Negatives and Positives
- Negatives that are classified as negatives – True Negatives (TN)
- Positives that are classified as positives – True Positives (TP)
- Negatives that are classified as positives – False Positives (FP)
- Positives that are classified as negatives – False Negatives (FN)

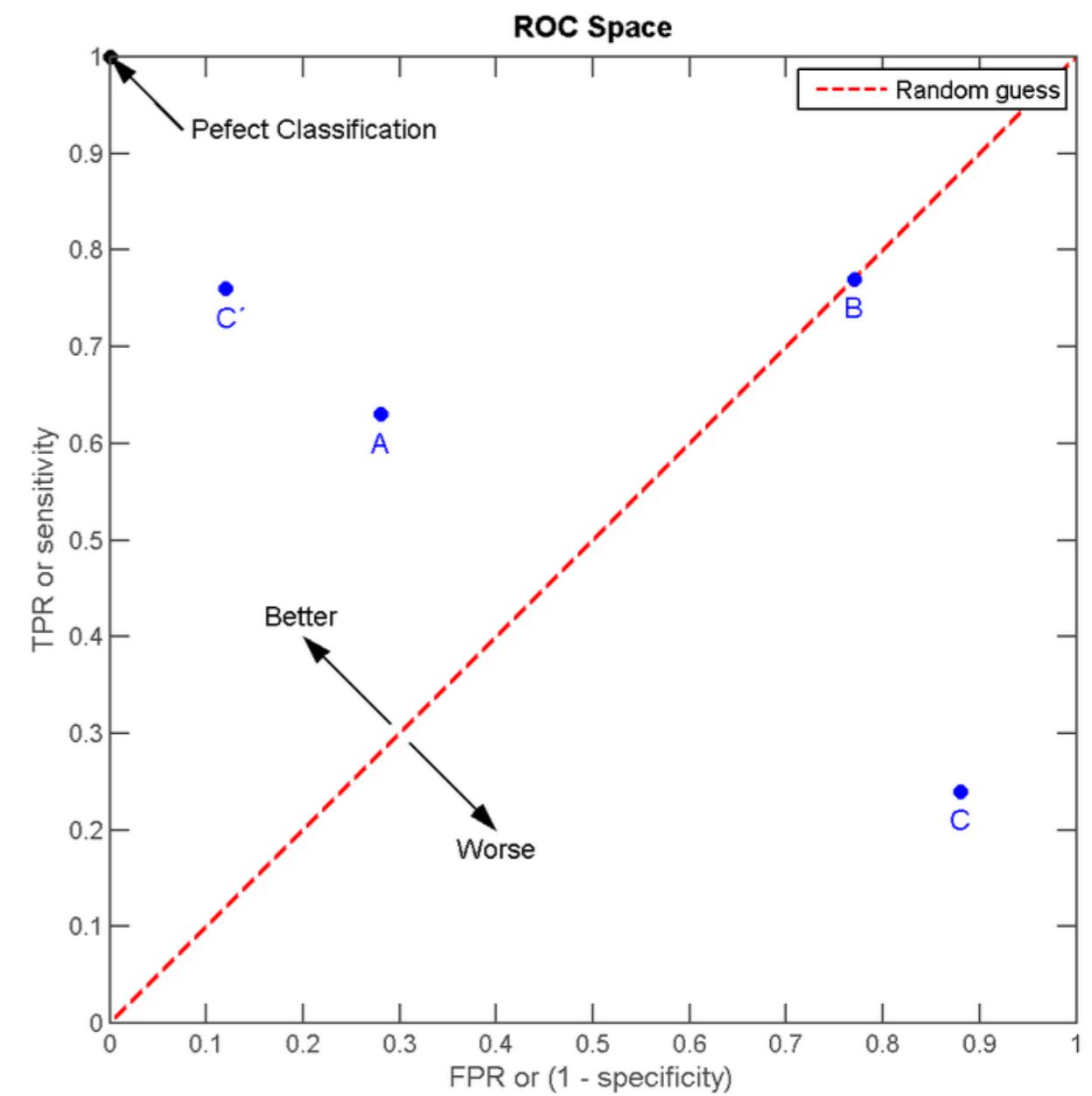
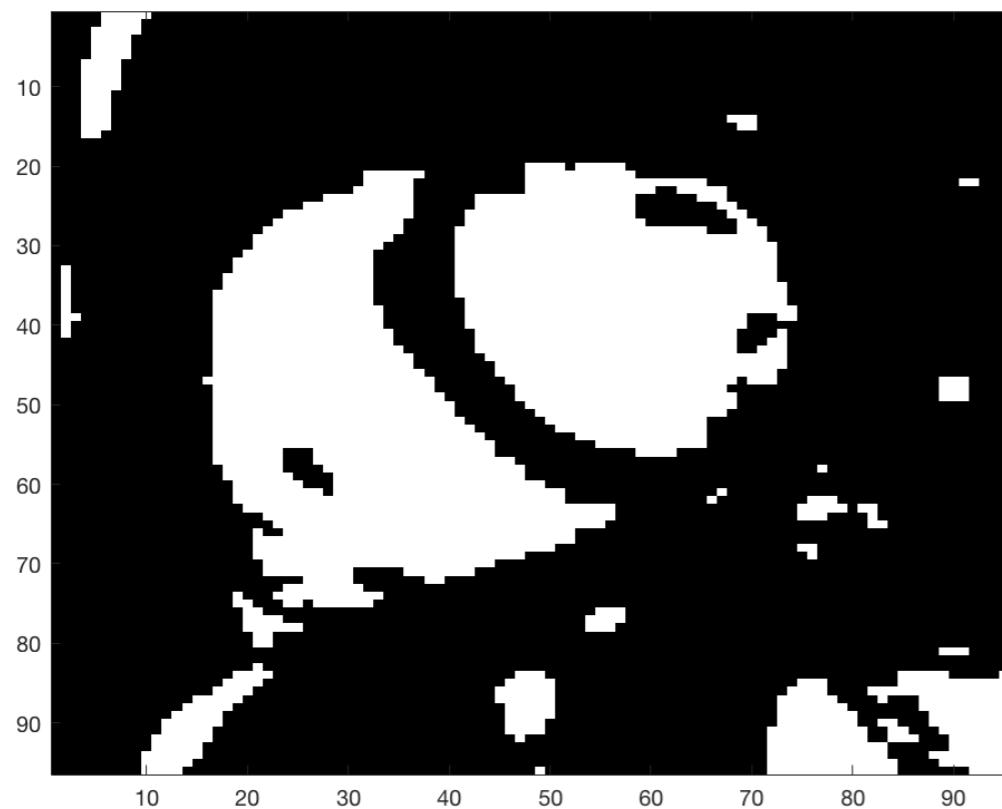
- False Positive Rate TPR = $FP/(FP+TN)$ -> x-axis
- True Positive Rate TPR = $TP/(TP+FN)$ -> y-axis



False Positives, False Negatives

ROC - Curve

- $FPR = FP/(FP+TN) \rightarrow x\text{-axis}$
- $TPR = TP/(TP+FN) \rightarrow y\text{-axis}$



Bayes Theorem

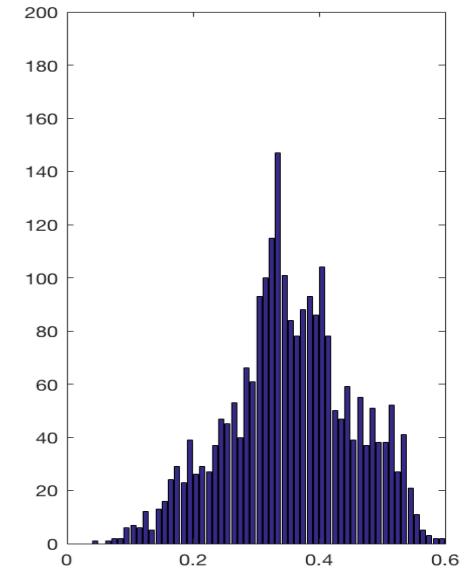
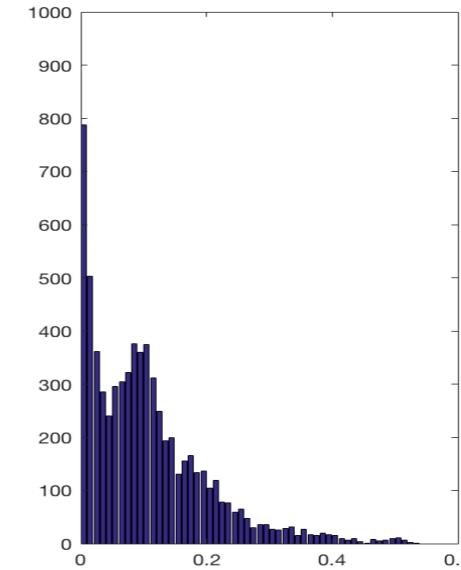
- Bayes Theorem

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

- Interpret P as probabilities, e.g. If X and Y are discrete
- Interpret P as probability density functions, e.g. If X and/or Y are continuous stochastic variable,

$$P(Y = y|X = x) = \frac{f_X(x|Y = y)P(Y = y)}{f_X(x)}$$

$$f_Y(y|X = x) = \frac{f_X(x|Y = y)f_Y(y)}{f_X(x)}$$





Sky

Castle

Grass

From 1 channel to 3
Colour images

Binning in higher dimensions

The curse of dimensionality

- 10 bins in one dimension -> 10 bins
- 10 bins in two dimensions -> 100 bins
- 10 bins in three dimensions -> 1000 bins
- 10 bins in 30 dimensions -> 1 000 000 000 000 000 000 000 000 bins
- 10 bins in 128 dimensions -> 10^{128} bins
- *"Many algorithms that work fine in low dimensions become intractable when the input is high-dimensional."*, Bellman, 1961.



Richard Bellman



LUND
UNIVERSITY

Clustering – adaptive binning

- Colour images. Pixels are RGB with 8 bits each. $2^{24} = 16777216$ types of pixels.
- Too many.
- Try to bin in a more clever way?
- Clustering



LUND
UNIVERSITY

Clustering: group together similar points and represent them with a single token

Key Challenges:

- 1) What makes two points/images/patches similar?
- 2) How do we compute an overall grouping from pairwise similarities?

Why do we cluster?

- **Summarizing data**
 - Look at large amounts of data
 - Patch-based compression or denoising
 - Represent a large continuous vector with the cluster number
- **Counting**
 - Histograms of texture, color, SIFT vectors
- **Segmentation**
 - Separate the image into different regions
- **Prediction**
 - Images in the same cluster may have the same labels

How do we cluster?

- K-means
 - Iteratively re-assign points to the nearest cluster center
- Agglomerative clustering
 - Start with each point as its own cluster and iteratively merge the closest clusters
- Mean-shift clustering
 - Estimate modes of pdf
- Spectral clustering
 - Split the nodes in a graph based on assigned links with similarity weights

Clustering for Summarization

Goal: cluster to minimize variance in data given clusters

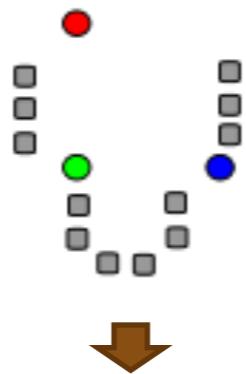
- Preserve information

$$\mathbf{c}^*, \boldsymbol{\delta}^* = \operatorname{argmin}_{\mathbf{c}, \boldsymbol{\delta}} \frac{1}{N} \sum_j^K \sum_i \boldsymbol{\delta}_{ij} (\mathbf{c}_i - \mathbf{x}_j)^2$$

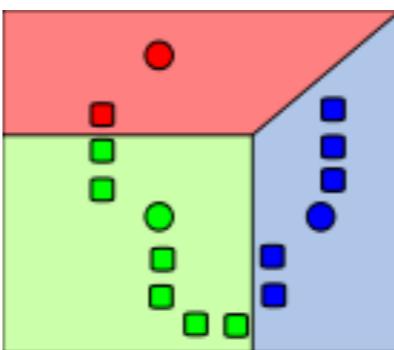
Cluster center Data
 ↑
 Whether \mathbf{x}_j is assigned to \mathbf{c}_i

K-means algorithm

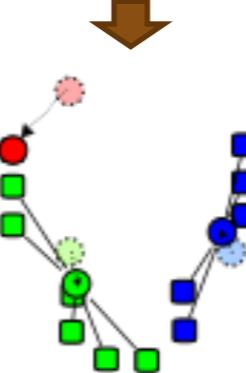
1. Randomly select K centers



2. Assign each point to nearest center

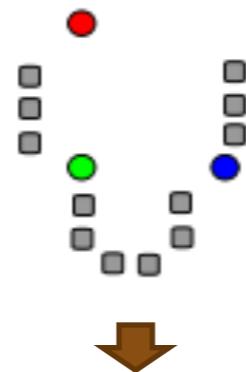


3. Compute new center (mean) for each cluster

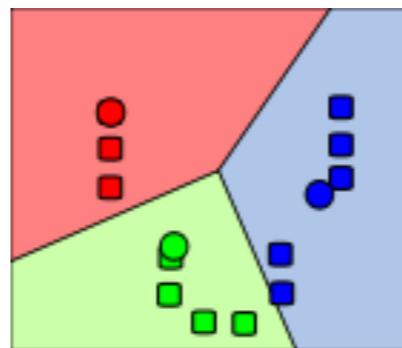


K-means algorithm

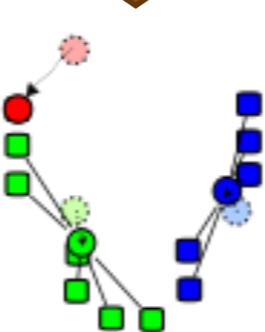
1. Randomly select K centers



2. Assign each point to nearest center



3. Compute new center (mean) for each cluster



Back to 2

K-means

1. Initialize cluster centers: \mathbf{c}^0 ; t=0

2. Assign each point to the closest center

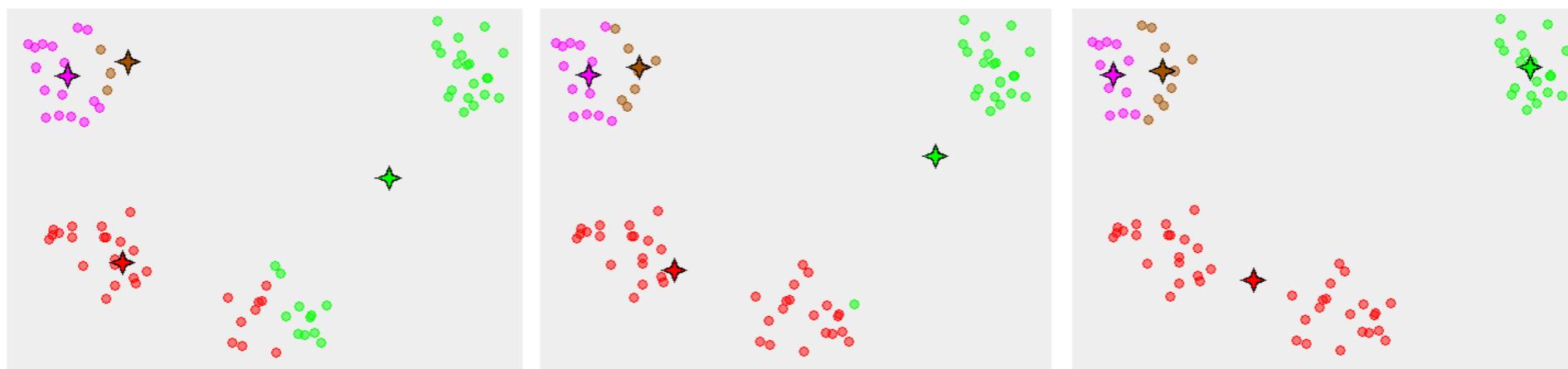
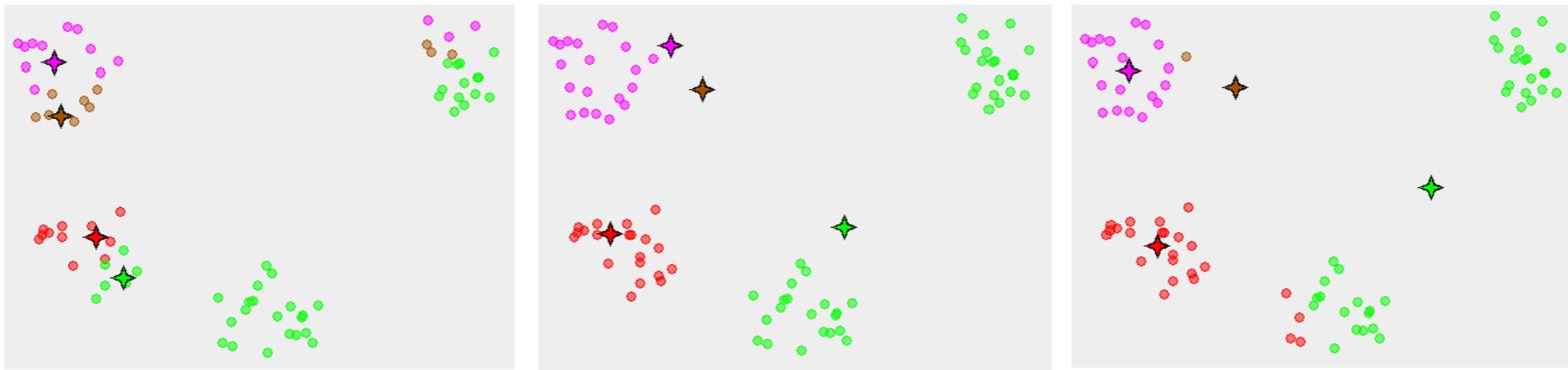
$$\boldsymbol{\delta}^t = \operatorname{argmin}_{\boldsymbol{\delta}} \frac{1}{N} \sum_j^K \sum_i^K \delta_{ij} (\mathbf{c}_i^{t-1} - \mathbf{x}_j)^2$$

3. Update cluster centers as the mean of the points

$$\mathbf{c}^t = \operatorname{argmin}_{\mathbf{c}} \frac{1}{N} \sum_j^K \sum_i^K \delta_{ij}^t (\mathbf{c}_i - \mathbf{x}_j)^2$$

4. Repeat 2-3 until no points are re-assigned (t=t+1)

K-means converges to a local minimum



K-means: design choices

- Initialization
 - Randomly select K points as initial cluster center
 - Or greedily choose K points to minimize residual
- Distance measures
 - Traditionally Euclidean, could be others
- Optimization
 - Will converge to a *local minimum*
 - May want to perform multiple restarts

Clustering

Goal: Partition a set on n feature vectors with d components

$$x_1, \dots, x_n$$

i.e. $x_i \in \mathbb{R}^d$, in groups (clusters) such that all examples in the same group are similar.

Clustering

Input: n examples x_1, \dots, x_n .

Output: A mapping

$$c : \{1, \dots, n\} \mapsto \{1, \dots, k\}.$$

Example: **K-means algorithm**

Choose k cluster centres m_1, \dots, m_k and a clustering function c that minimises

$$f(c, m) = \sum_{i=1}^n \|x_i - m_{c(i)}\|^2.$$

K-means

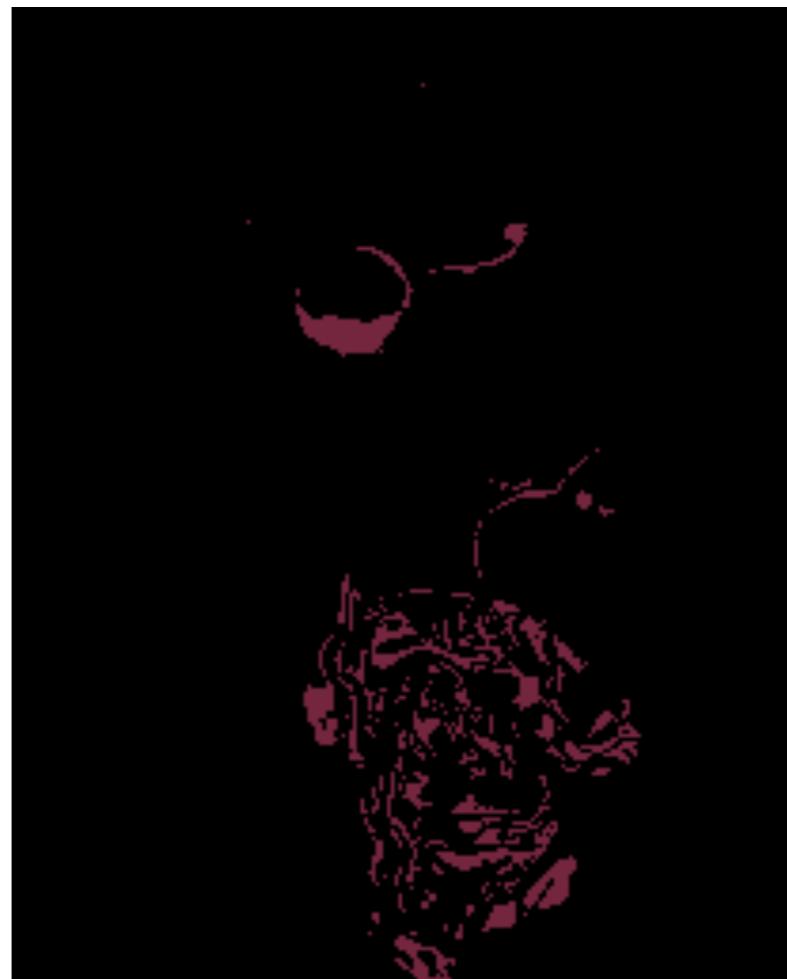
The problem

$$\min_{c,m} f(c, m) = \min_{c,m} \sum_{i=1}^n ||x_i - m_{c(i)}||^2$$

is a non-linear optimization problem that can be solved with many methods. However, most only give a local optima.



K-means using
color alone,
11 segments.





K-means using colour and position, 20 segments



K-means clustering using intensity or color

Image



Clusters on intensity



Clusters on color



How to choose the number of clusters?

- Minimum Description Length (MDL)
principal for model comparison
- Minimize Schwarz Criterion
 - also $\text{Distortion} + \lambda (\# \text{parameters}) \log R$ C)

$$= \text{Distortion} + \lambda m k \log R$$

m = #dimensions

k = #Centers

R = #Records

How to evaluate clusters?

- Generative
 - How well are points reconstructed from the clusters?
- Discriminative
 - How well do the clusters correspond to labels?
 - Purity
 - Note: unsupervised clustering does not aim to be discriminative

How to choose the number of clusters?

- Validation set
 - Try different numbers of clusters and look at performance
 - When building dictionaries (discussed later), more clusters typically work better

K-means demos

General

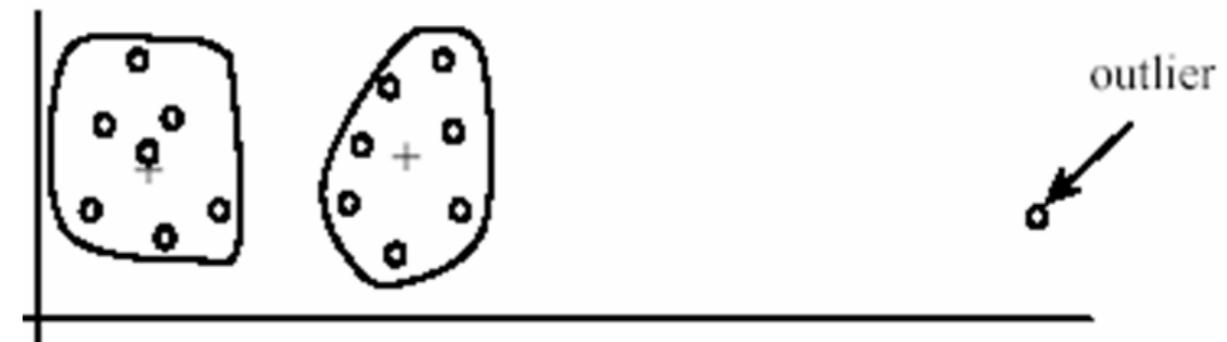
<http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html>

Colour Clustering

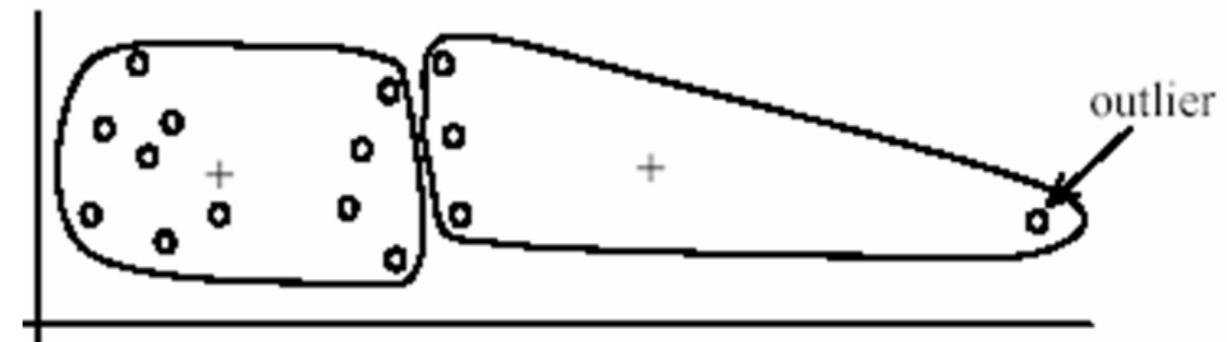
<http://www.cs.washington.edu/research/imagedatabase/demo/kmcluster/>

K-Means pros and cons

- Pros
 - Finds cluster centers that minimize conditional variance (good representation of data)
 - Simple and fast*
 - Easy to implement
- Cons
 - Need to choose K
 - Sensitive to outliers
 - Prone to local minima
 - All clusters have the same parameters (e.g., distance measure is non-adaptive)
 - *Can be slow: each iteration is $O(KNd)$ for N d-dimensional points
- Usage
 - Rarely used for pixel segmentation

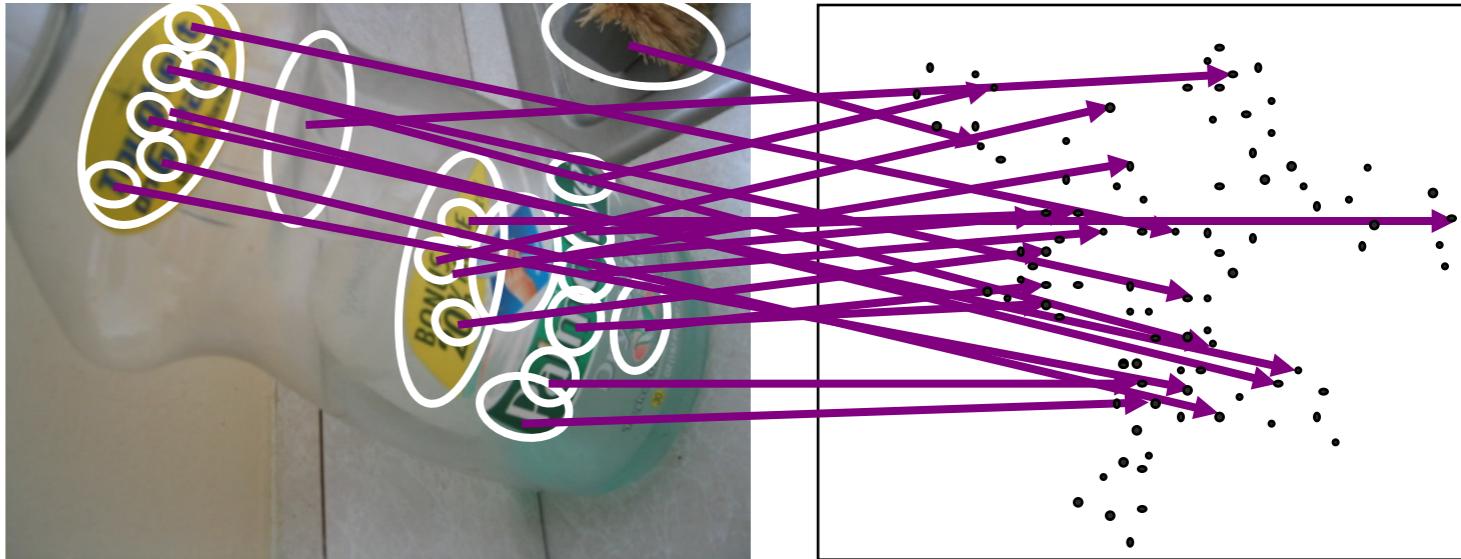


(B): Ideal clusters

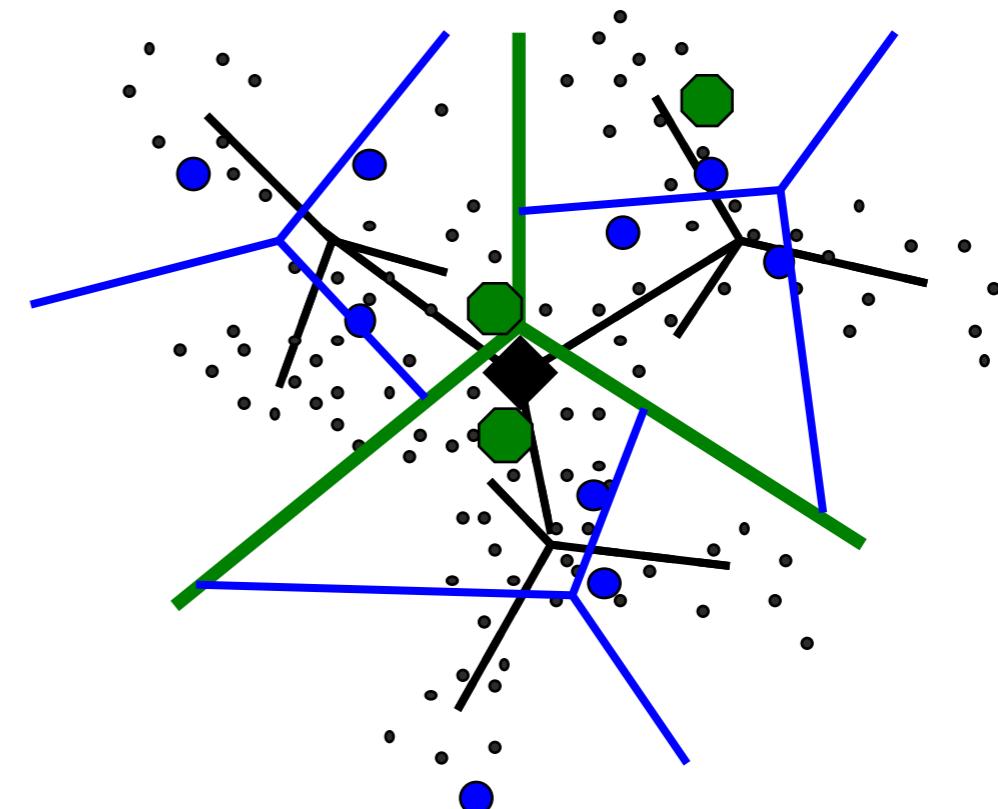


Building Visual Dictionaries

1. Sample patches from a database
 - E.g., 128 dimensional SIFT vectors

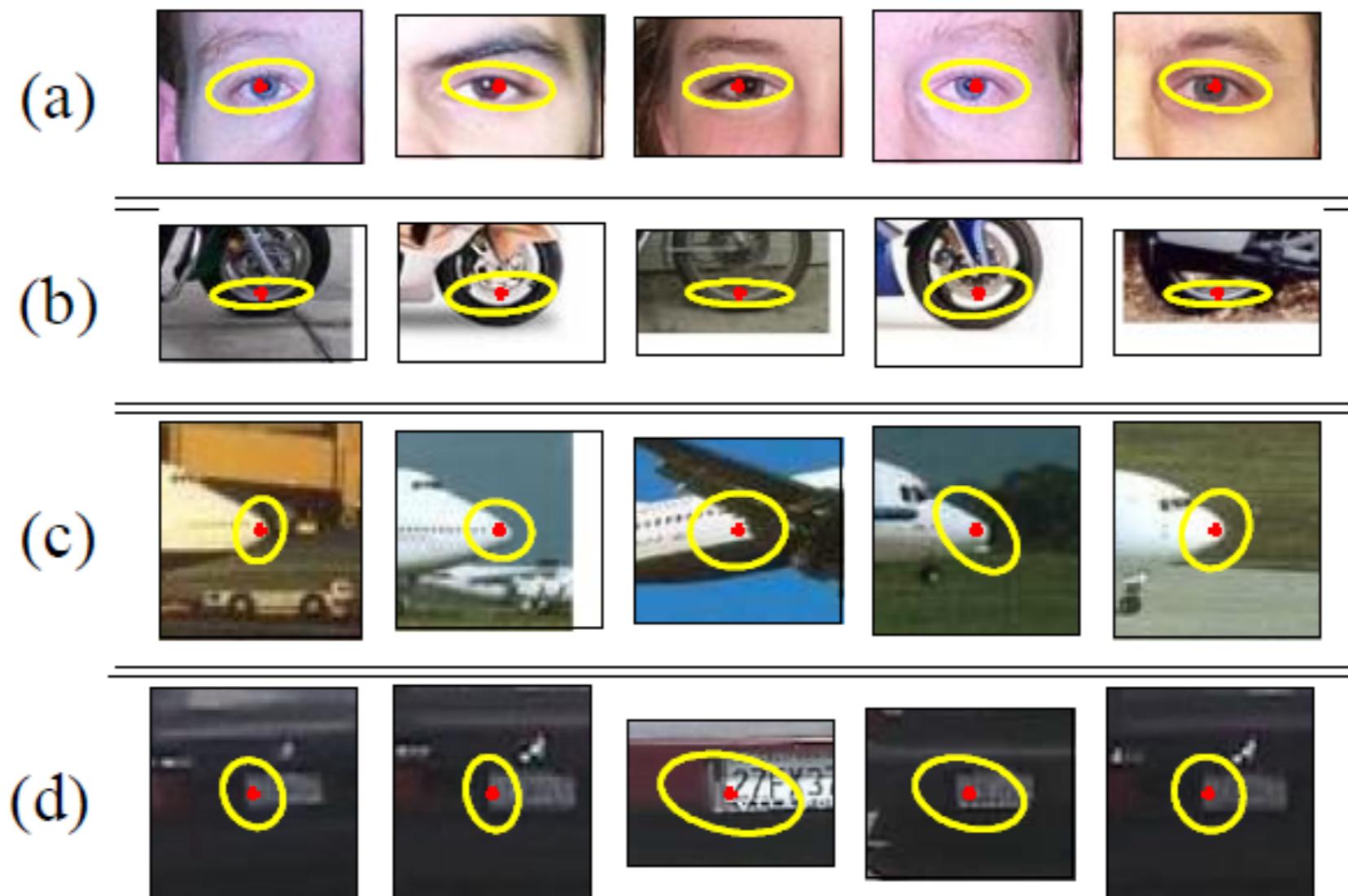


2. Cluster the patches
 - Cluster centers are the dictionary



3. Assign a codeword (number) to each new patch, according to the nearest cluster

Examples of learned codewords



Most likely codewords for 4 learned “topics”
EM with multinomial (problem 3) to get topics

Example:

Colour pixel classification

- Use clustering to assign codewords (bin nr) x 1-10 for each pixel
- Estimate measurement probabilities $p(x|y)$ for each class y (1-grass, 2-castle, 3-sky) and each bin x .
- Use Bayes theorem to calculate $p(y|x)$ for each pixel,
- Classify according to maximum a posteriori probability



Example: Colour pixel classification



Cluster nr: 1 out of 10



Cluster nr: 2 out of 10



Cluster nr: 3 out of 10



Cluster nr: 4 out of 10



Cluster nr: 5 out of 10



Cluster nr: 6 out of 10



Cluster nr: 7 out of 10



Cluster nr: 8 out of 10



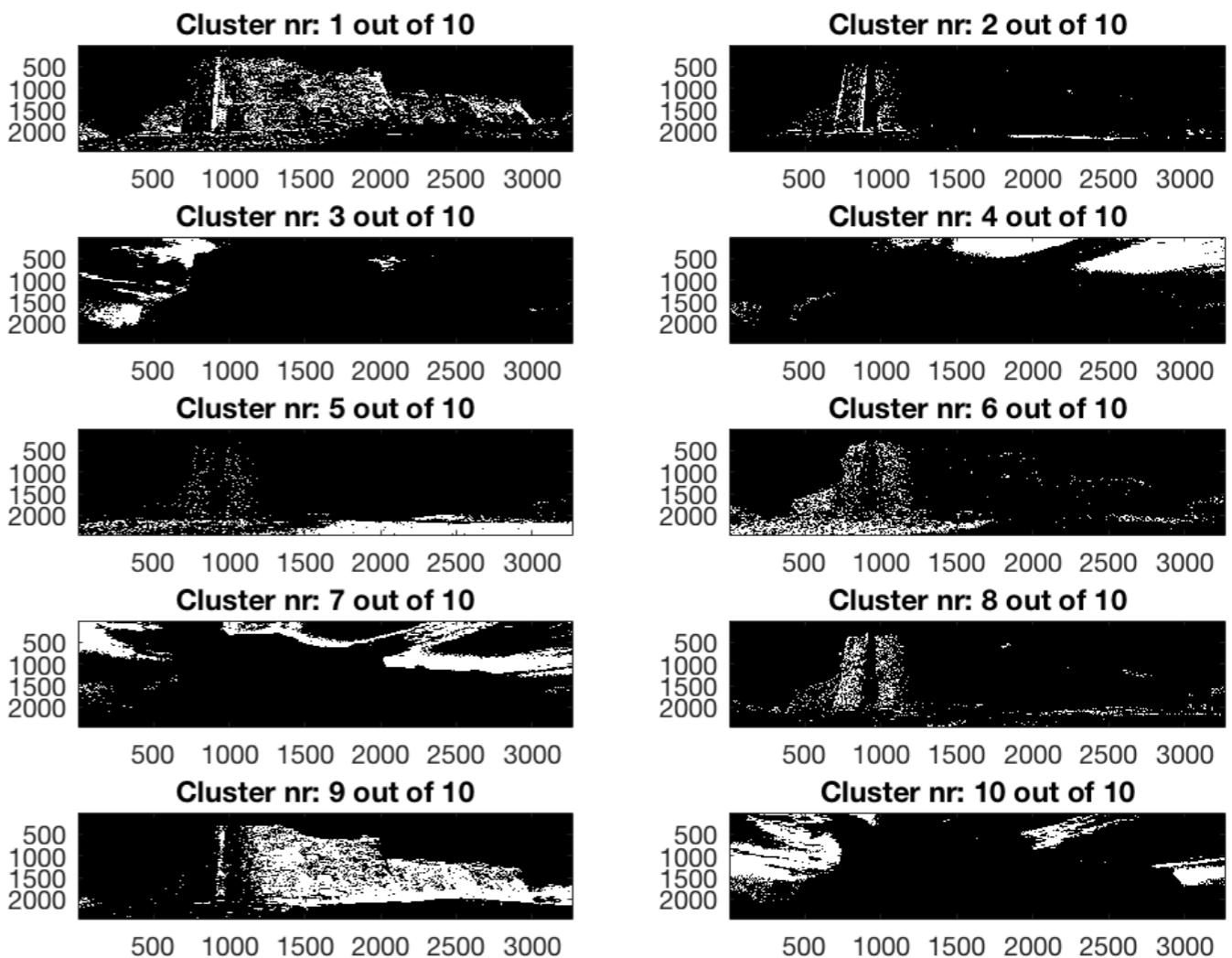
Cluster nr: 9 out of 10

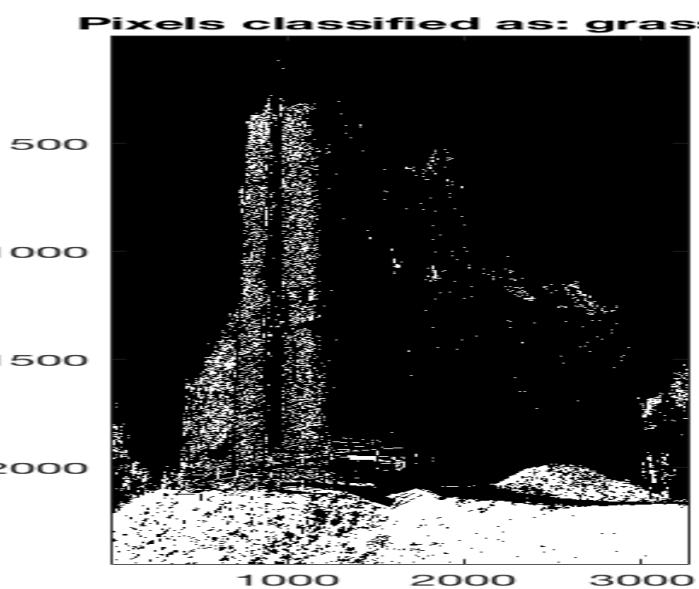
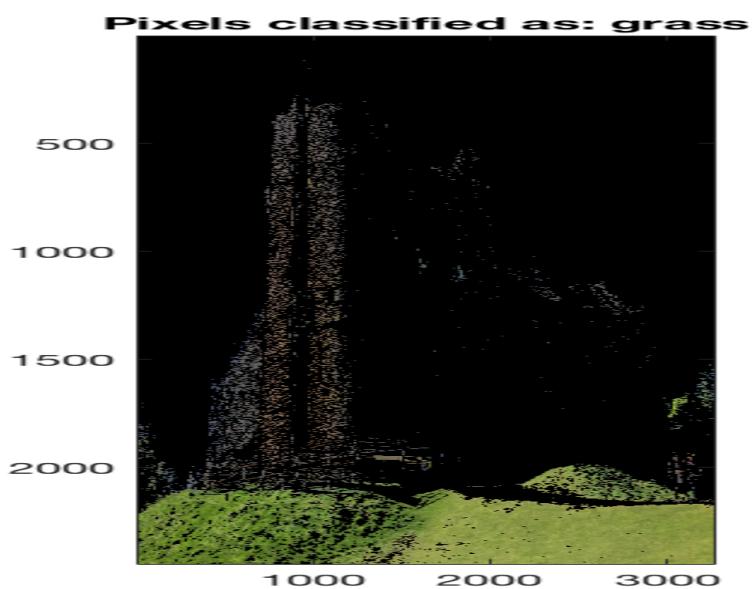
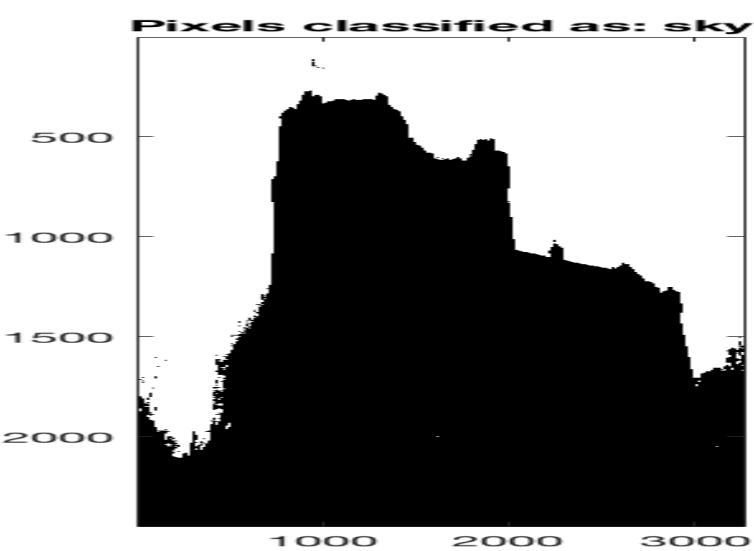
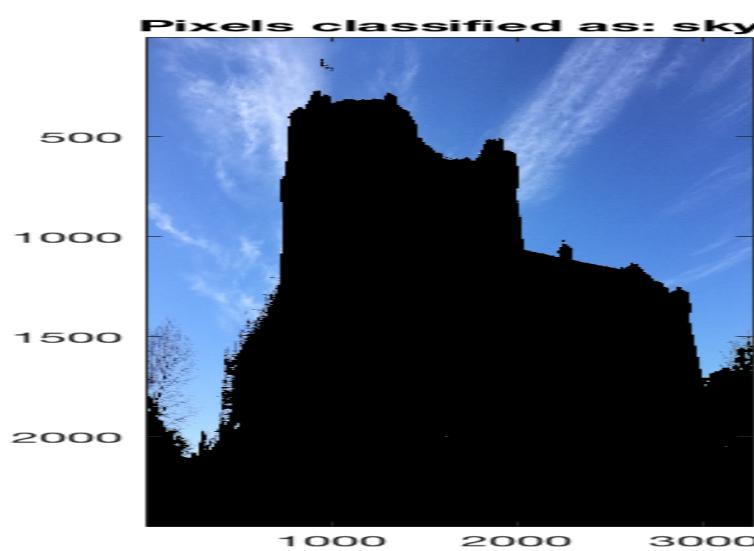
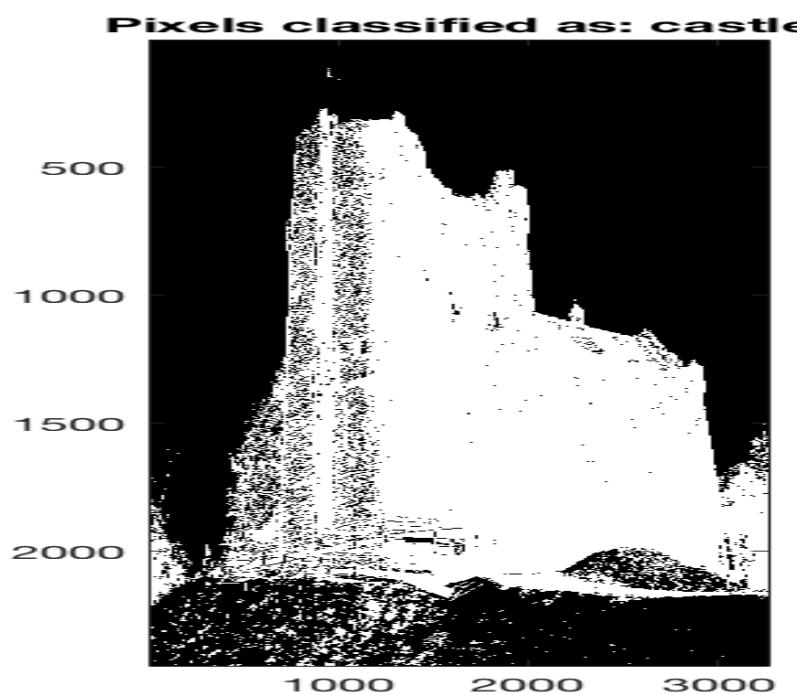


Cluster nr: 10 out of 10



Example: Colour pixel classification



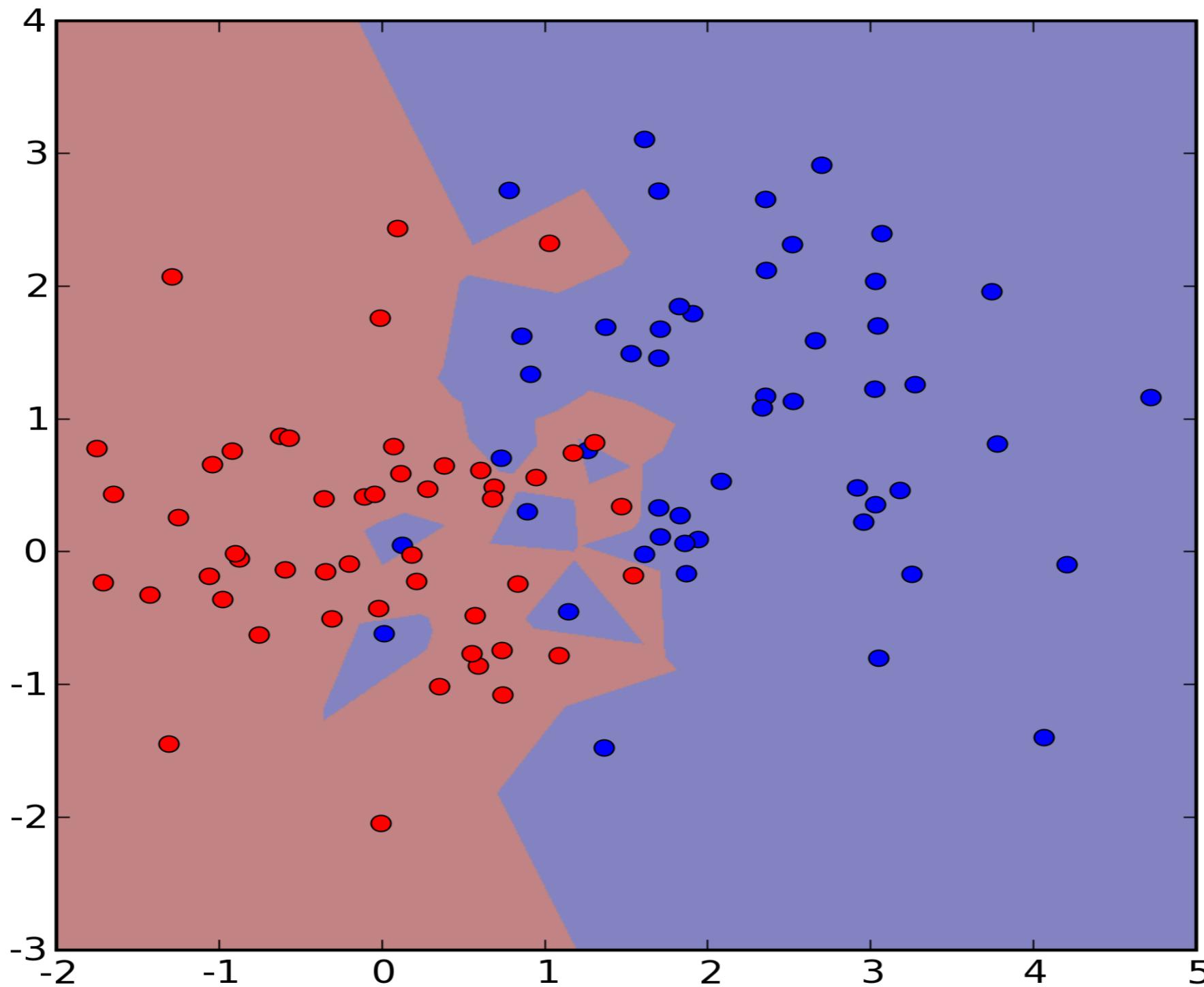


Nearest Neighbour Classification

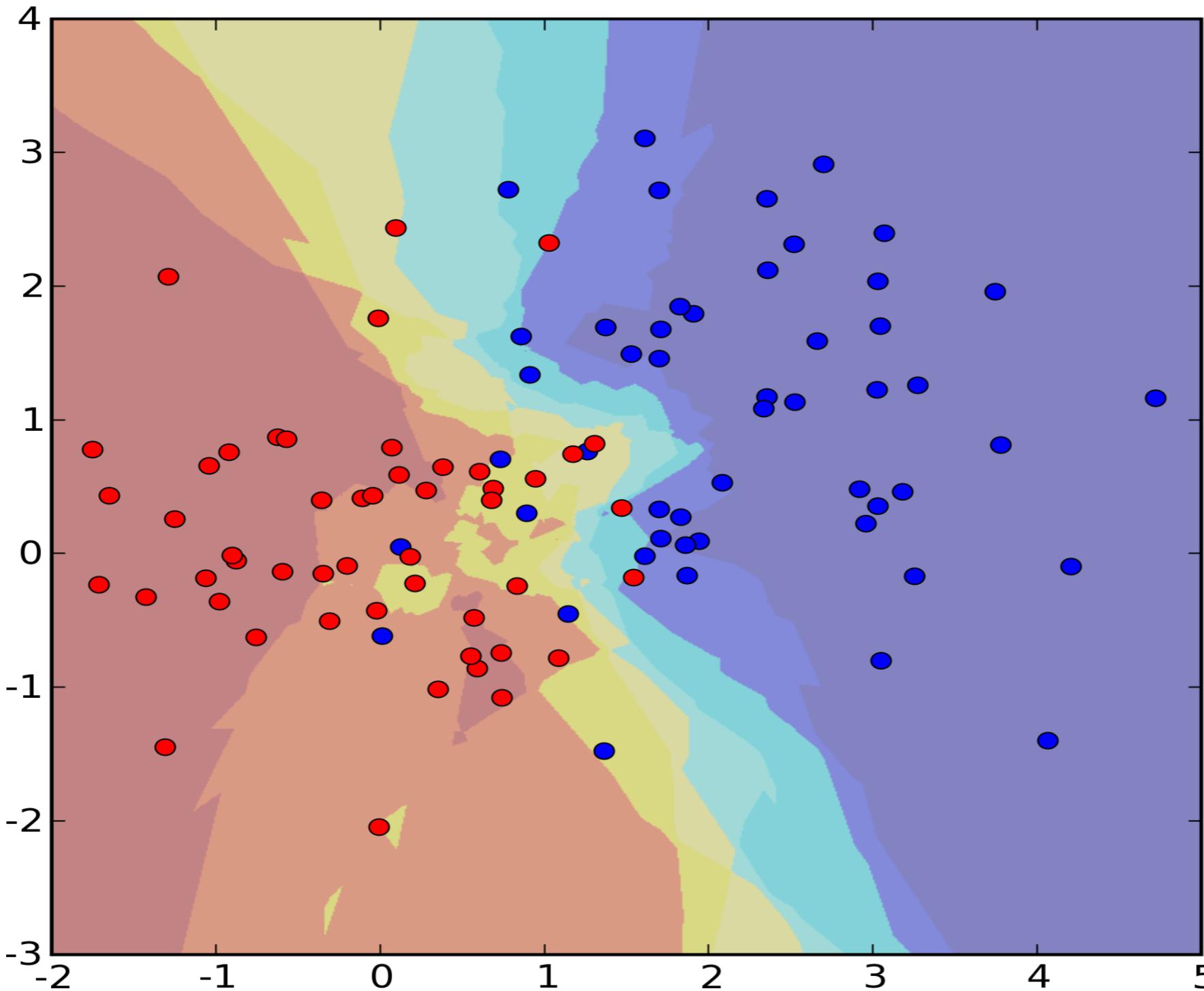
NN and K-NN

- Classify using training data (x_i, y_i)
- NN: Use the label of the nearest neighbour
- KNN: Use the label of the majority of the k nearest neighbours
- Regression: Use the average of the value of the k nearest neighbours
- Easy to implement and understand
- Can use arbitrary distance functions between images
- Converges to the optimum
- Slow when using lots of data, need to store all training data, not smooth regression

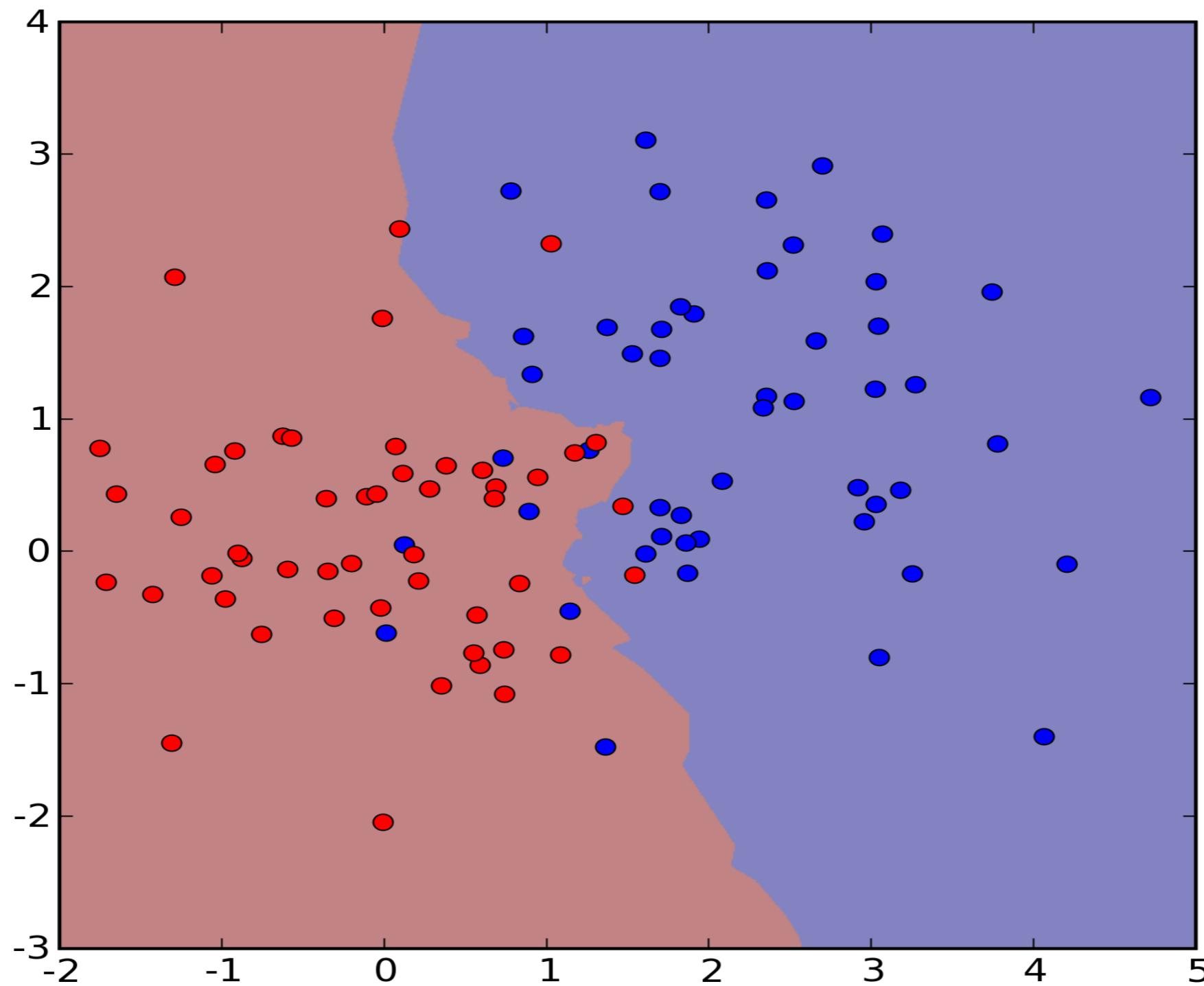
Nearest Neighbour Classification (discussion)



7 Nearest Neighbour Classification



7 Nearest Neighbour Classification



Nearest Neighbour Classification

NN and K-NN

- Training is easy, just store the training data $T = \{(x_1, y_1), \dots, (x_N, y_N)\}$
- Works in any dimension
- Works for regression also: Use the average of the value of the k nearest neighbours
- Easy to implement and understand
- Can use arbitrary distance functions between images
- Converges to the optimum
- Slow when using lots of data,
- Need to store all training data
- Not smooth regression

Parametric density estimation

Plug-in Classifier

- **Parametric density estimation**
- (compare with Nonparametric density estimation)
 - Parametric – fixed nr of parameters
 - Nonparametric – nr of parameters grow with training data
- Plug-in classifier, i.e. plug-in the estimated densities in Bayes rule
- Classification



Parametric density estimation

Plug-in Classifier

- Bayes Theorem

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

- Interpret P as probabilities, e.g. If Y – discrete
- Interpret P as probability density functions, e.g. If X and/or Y are continuous stochastic variable,

$$P(Y = y|X = x) = \frac{f_X(x|Y = y)P(Y = y)}{f_X(x)}$$

$$f_Y(y|X = x) = \frac{f_X(x|Y = y)f_Y(y)}{f_X(x)}$$

Parametric density estimation

Example: Pixel classification. Gray-scale image, easier to plot 1D distributions.
Classify 'Castle' vs ('sky' and 'grass')



Parametric density estimation

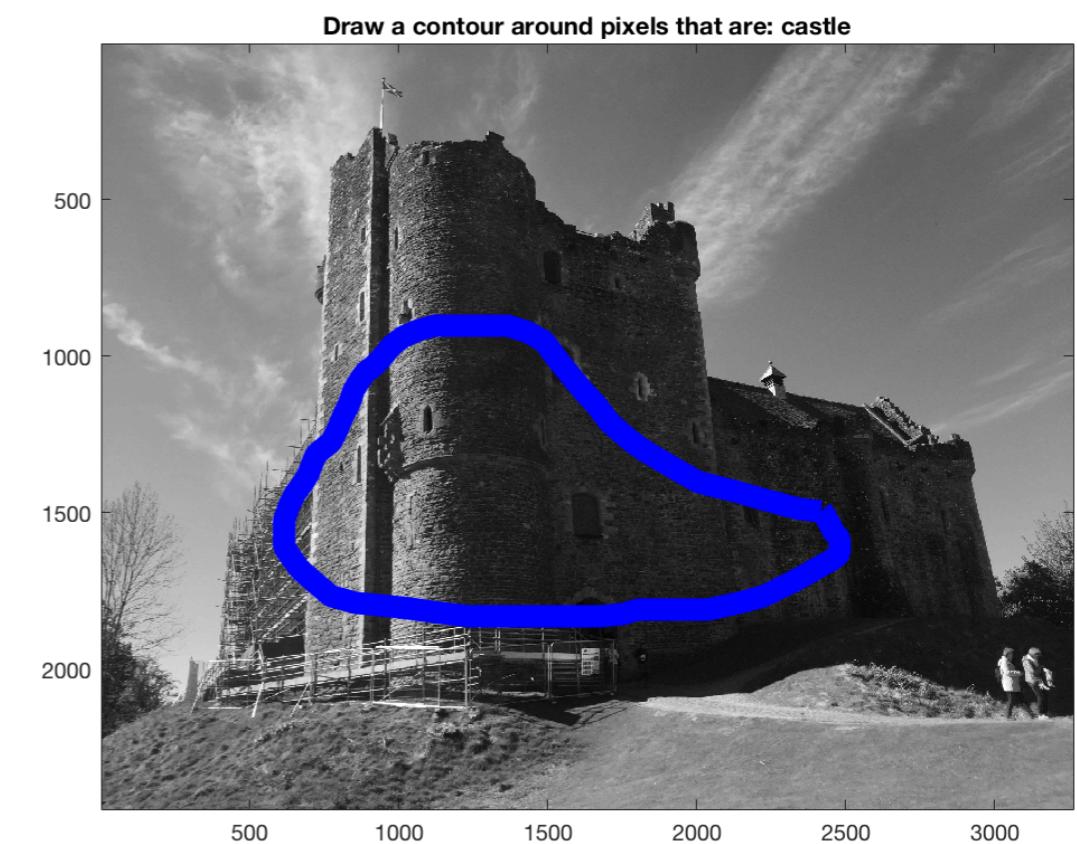
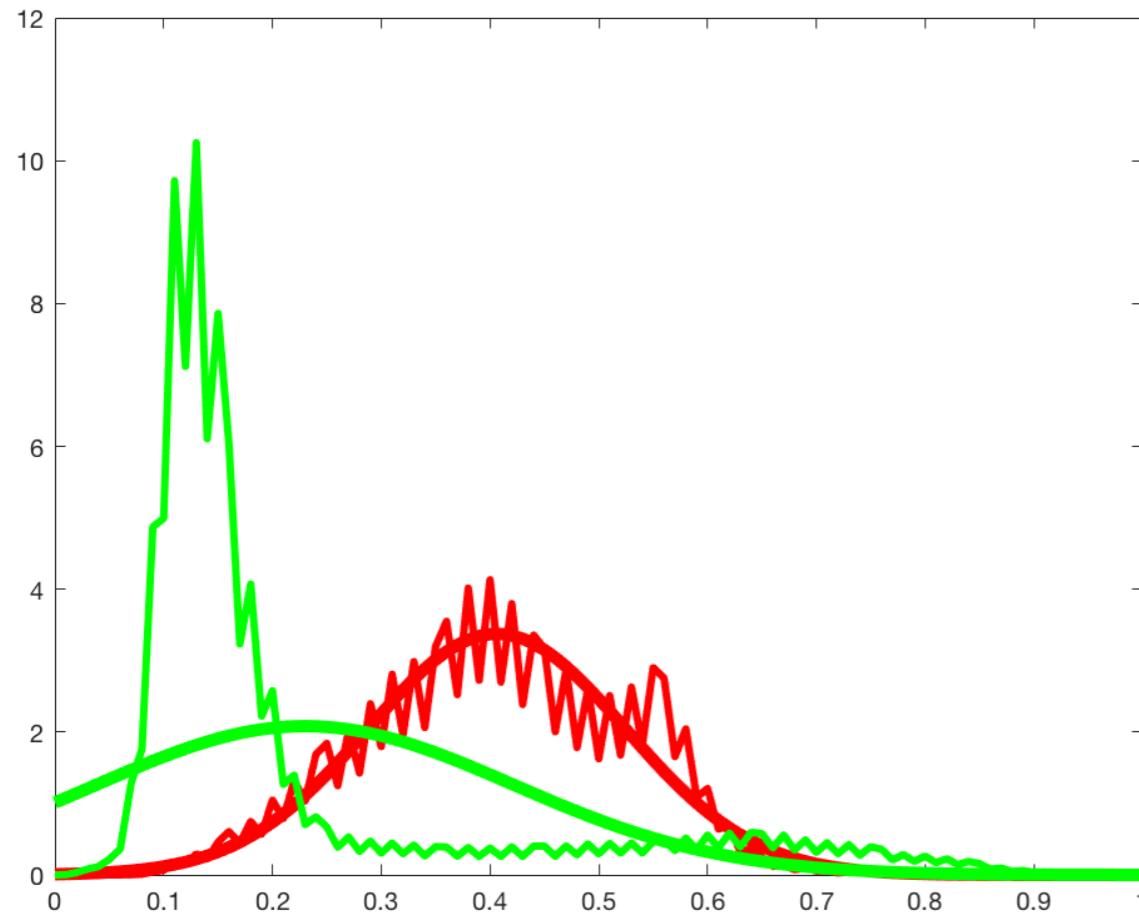
'Castle' vs ('sky' and 'grass')

Since it is 1D. Easy to both binning and parametric density estimation

Assume normal distribution.

Two parameters, mean m and standard deviation σ .

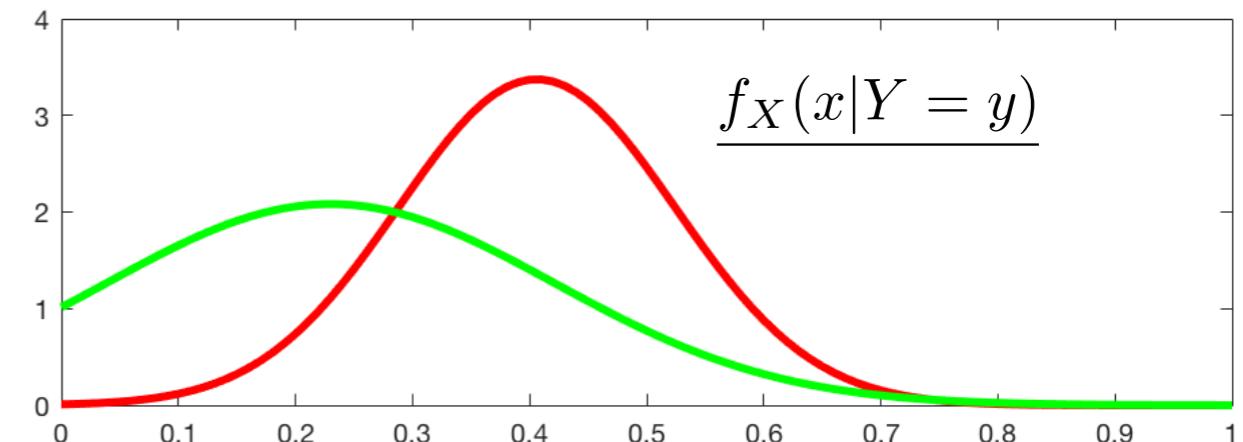
Estimate using maximum likelihood method.



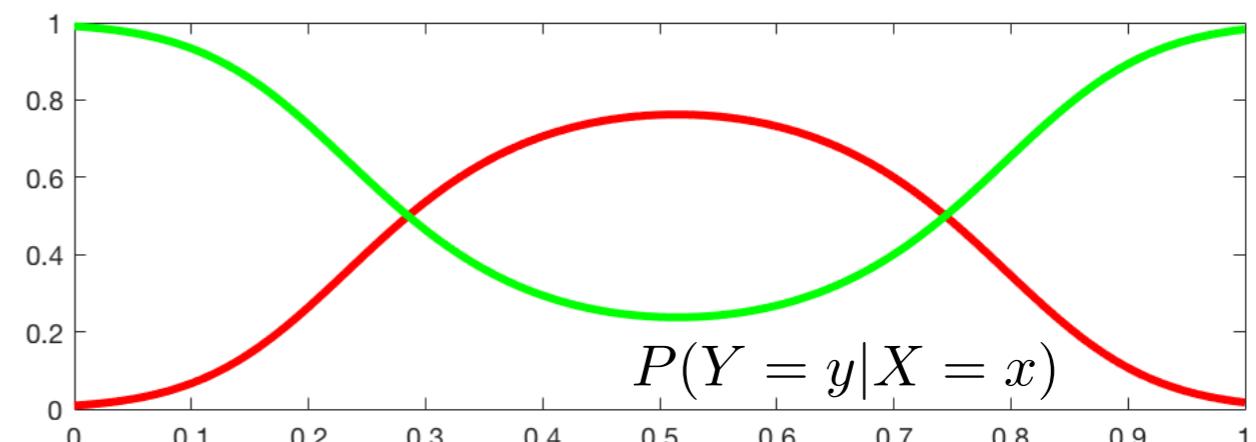
Parametric density estimation

1. Parametric density estimation – estimate m and sigma, (estimate or guess prior)
2. Plug-in classifier, i.e. plug-in the estimated densities (and priors) in Bayes rule

$$P(Y = y|X = x) = \frac{f_X(x|Y = y)P(Y = y)}{f_X(x)}$$



3. Classification



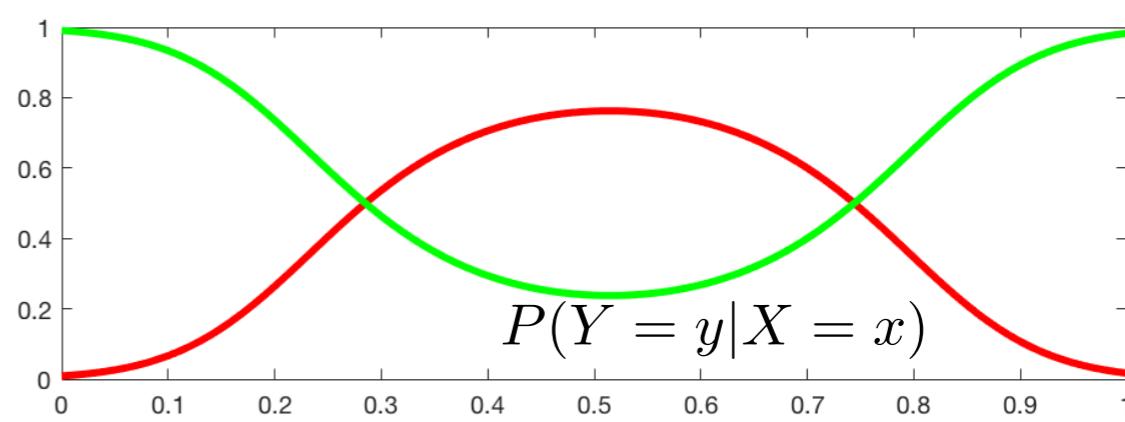
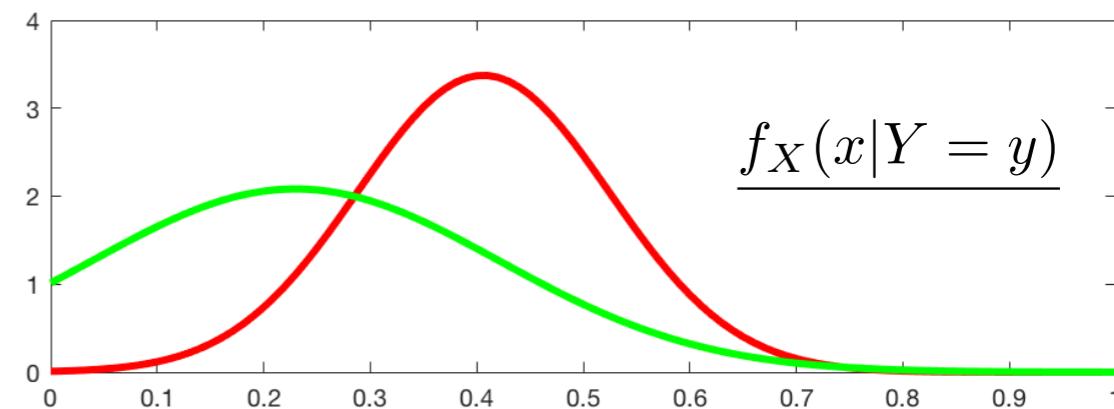
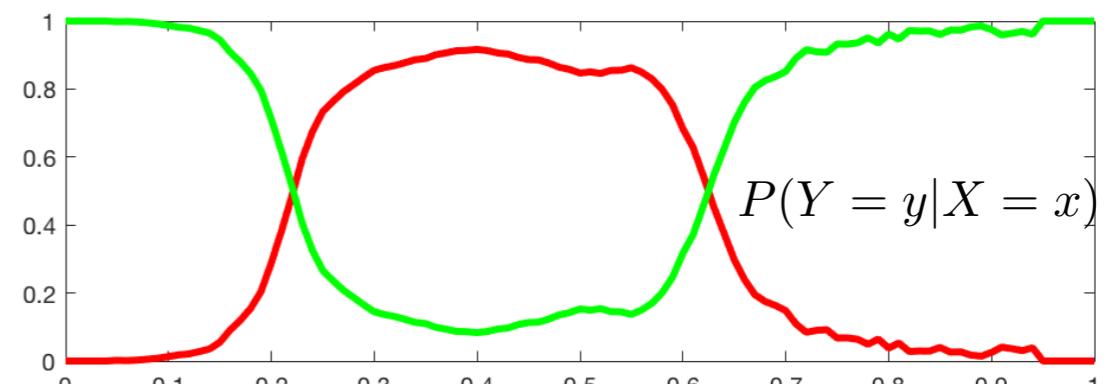
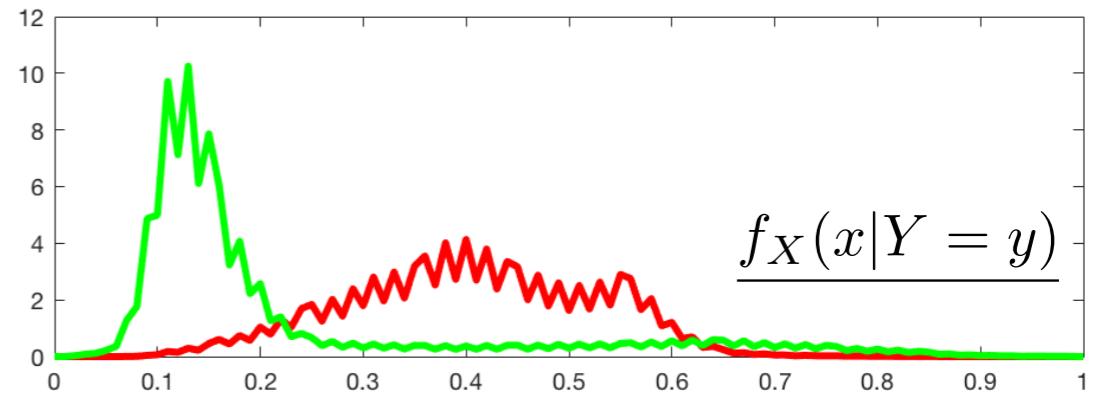
Compare binning Vs parametric

1. Binning

1. Works only in 1 or a few dimensions
2. Can be used in higher dimensions with adaptive binning (clustering, k-means)
3. Discontinuous

2. Parametric density estimation

1. Must guess density model
2. Fewer parameters to estimate
3. Fewer parameters to store
4. Can be smooth



Parametric density estimation

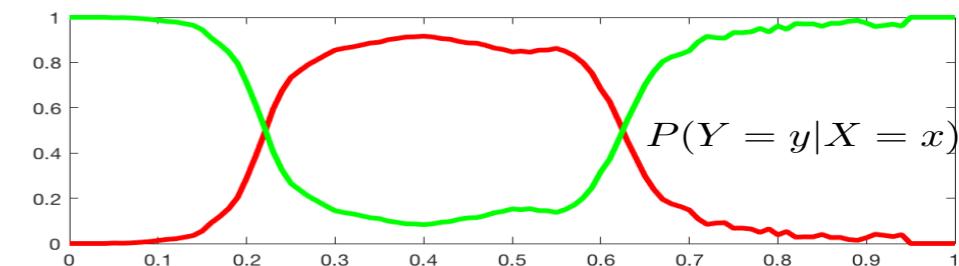
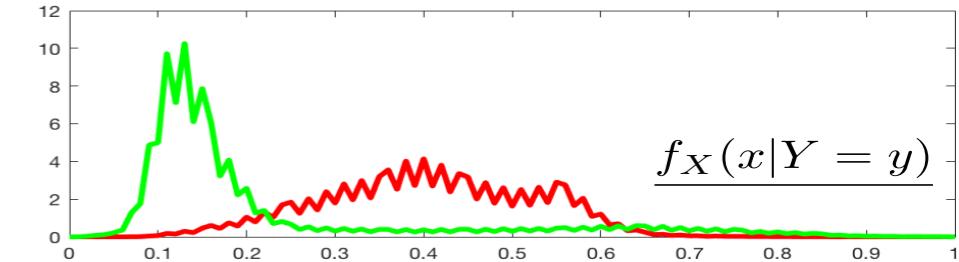
- Many interesting and useful parametric densities to choose from
- Estimation of parameters can be
 - easy (simple formulas) or
 - hard
 - involve convex optimization (no local optima, guaranteed results)
 - non-convex optimization (many local optima, no guarantee on finding the best optima).
- Fewer parameters to estimate
- Fewer parameters to store
- Can be smooth

Logistic regression

- Motivation
- In the end we are only interested in the posterior distribution

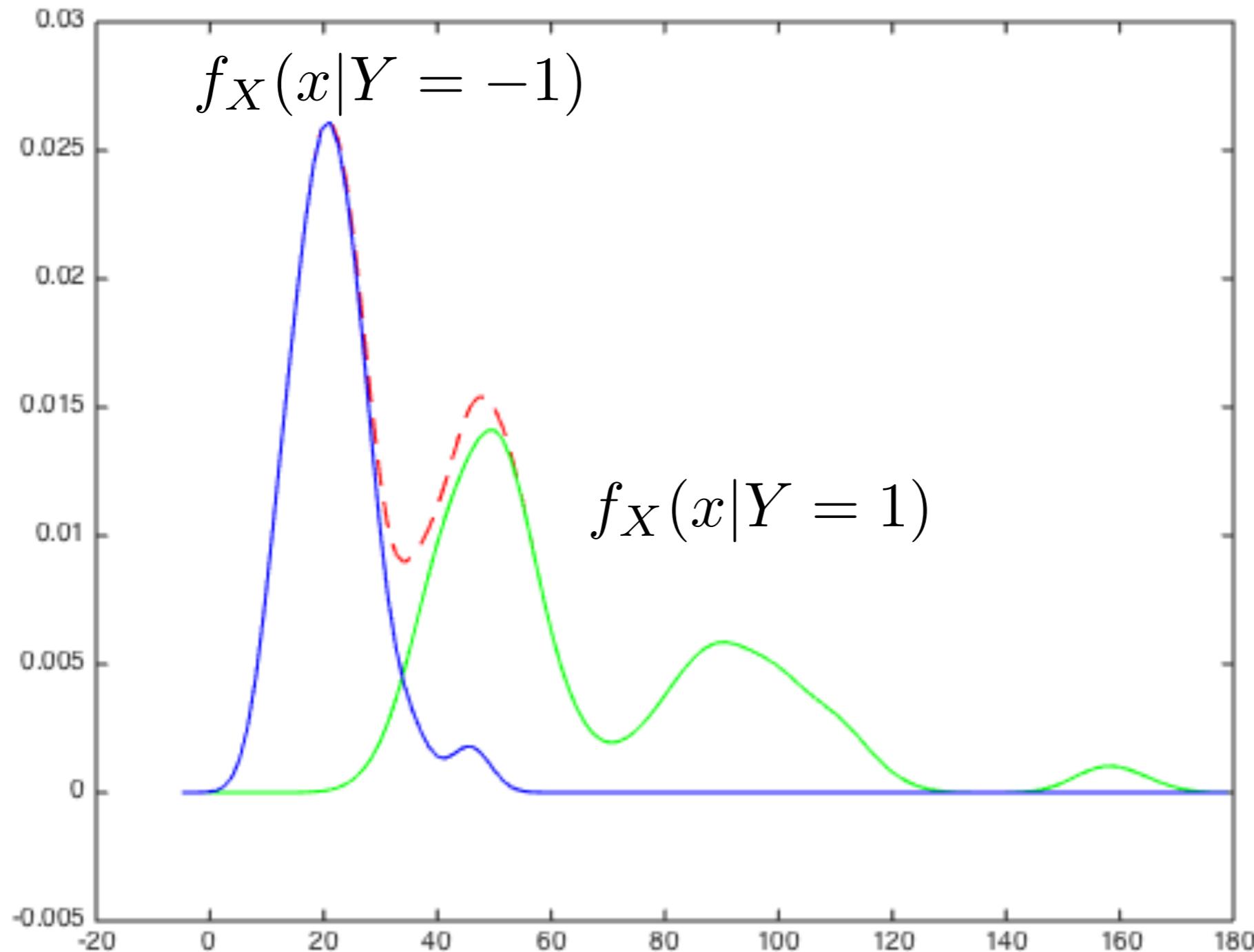
$$P(Y = y|X = x)$$

- Why not estimate this instead
- Skip the step of estimating the measurement densities
- Details far away from the transition points are uninteresting (perhaps)
- Notice that the posterior looks like a smoothed step function

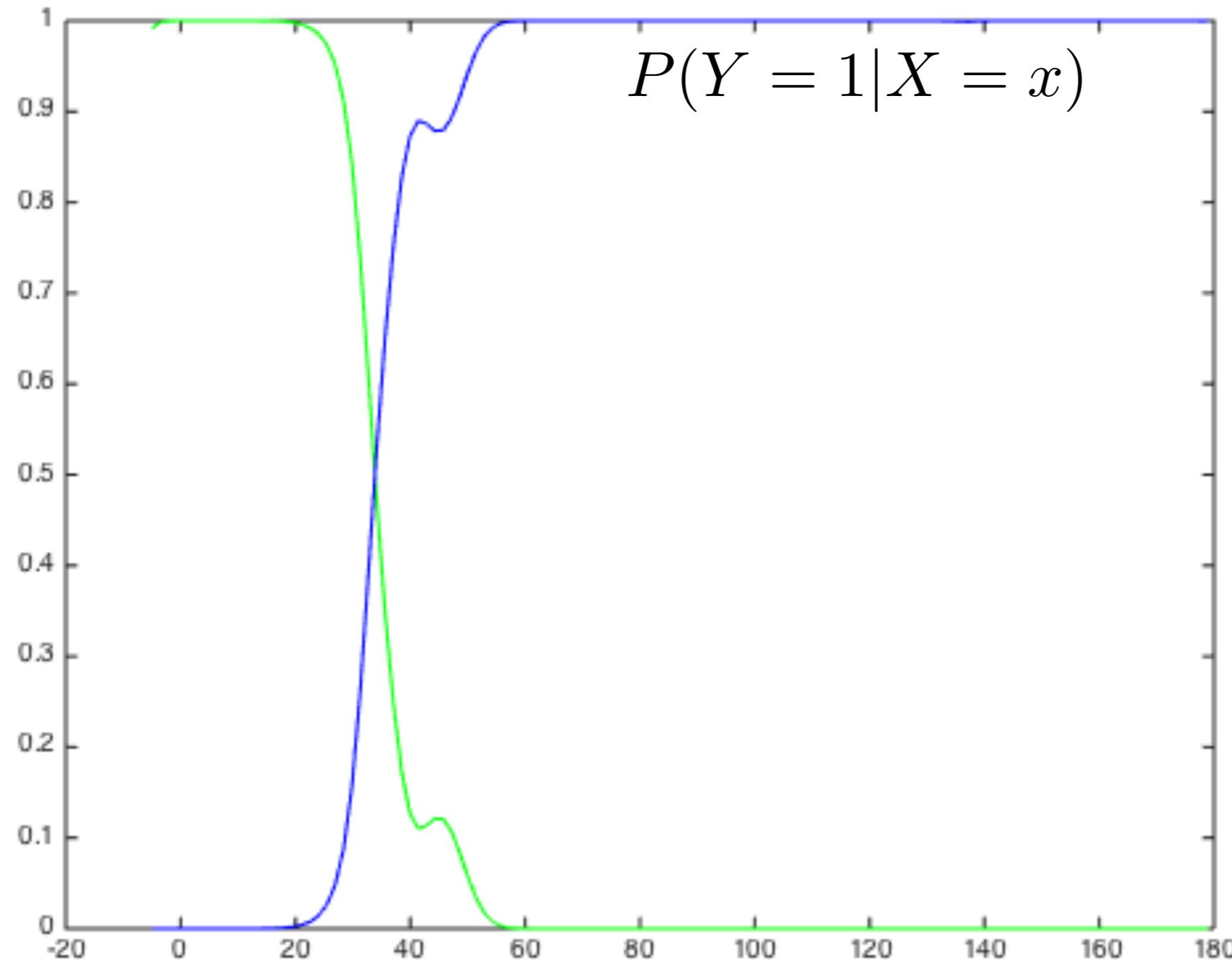


$$\underline{f_X(x|Y=y)}$$

Example $P(Y = y|X = x) = \frac{f_X(x|Y = y)P(Y = y)}{f_X(x)}$



Example $P(Y = y|X = x) = \frac{f_X(x|Y = y)P(Y = y)}{f_X(x)}$



Logistic regression

- Discuss ideas and derivations on blackboard
- z = simple function of x , e.g. Linear $z = w^T x + b$
- Output y = smooth threshold of z , for example

$$s(z) = \frac{1}{1 + e^{-z}}$$

- Notice that $s(z)$ looks like a typical $P(Y=1 | x)$ function

$$x \in R^d, w \in R^d, b \in R, f(x) = s(w^T x + b)$$

$$P(Y = 1 | x) = \frac{1}{1 + e^{-z}}$$

Derivation

- Estimate parameters

$$P(Y = 1|x) = \frac{1}{1 + e^{-z}}$$

$$P(Y = -1|x) = 1 - \frac{1}{1 + e^{-z}} = \frac{e^{-z}}{1 + e^{-z}} = \frac{1}{e^z + 1}$$

- For both cases we have

$$P(Y = y|x) = \frac{1}{1 + e^{-yz}}$$

- Calculate likelihood for training data

$$T = (x_1, y_1), \dots, (x_n, y_n)$$

Estimate parameters

- Parameters $\theta = (w, b)$

$$P(Y = y|x) = \frac{1}{1 + e^{-yz}}$$

$$T = (x_1, y_1), \dots, (x_n, y_n)$$

$$\begin{aligned} \log(P) &= \log\left(\prod_i P(Y = y_i | x_i, \theta)\right) \\ &\quad \sum_i \log\left(\frac{1}{1 + e^{y_i(w^T x_i + b)}}\right) \end{aligned}$$

$$\min_{\mathbf{w}} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^l \log(1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i}).$$

Logistic regression

- Linear logistic regression
- Estimate the posterior $P(Y = y|X = x)$

Standard
logistic function

$$s(z) = \frac{1}{1 + e^{-x}}$$

- As linear function followed by standard logistic function

$$s(w^T x + b)$$

- Convex optimization problem

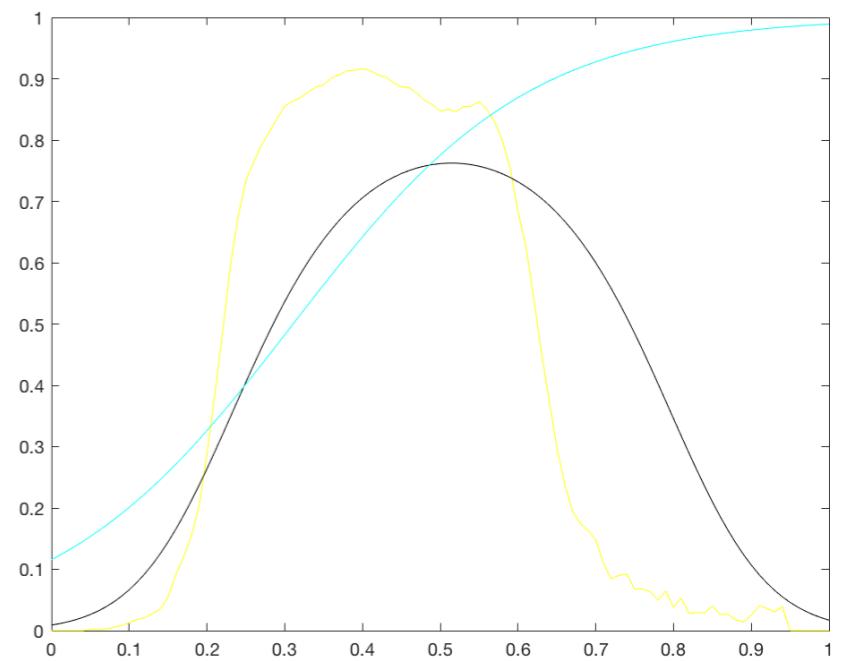
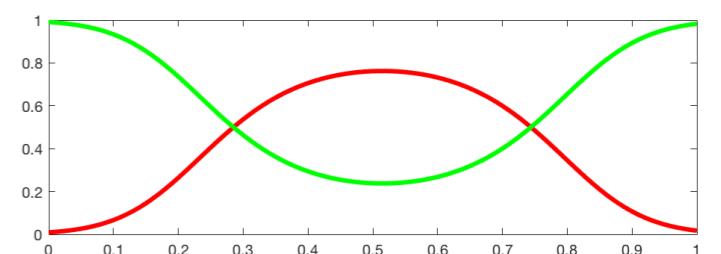
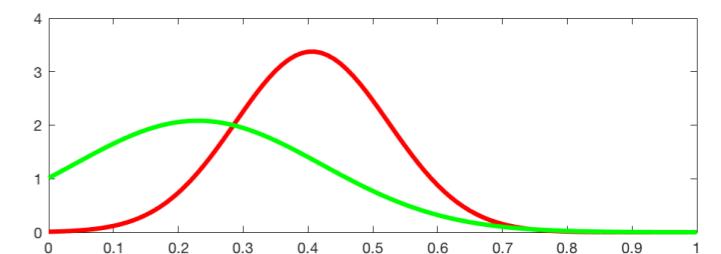
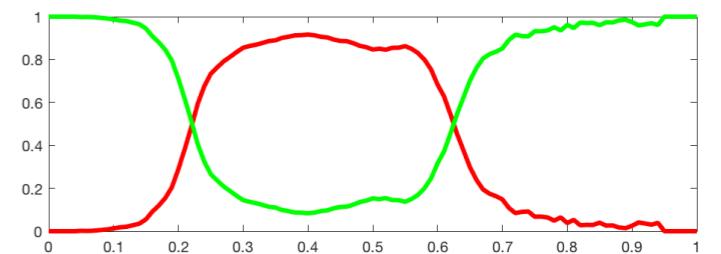
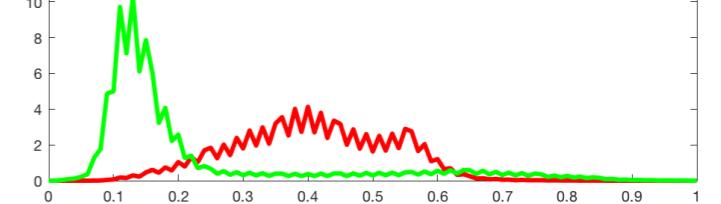
Logistic regression

- Estimate w and b
- Posterior distribution $P(Y = y|X = x)$
- Simple model $s(w^T x + b)$



Standard logistic function

$$s(z) = \frac{1}{1 + e^{-x}}$$



Review

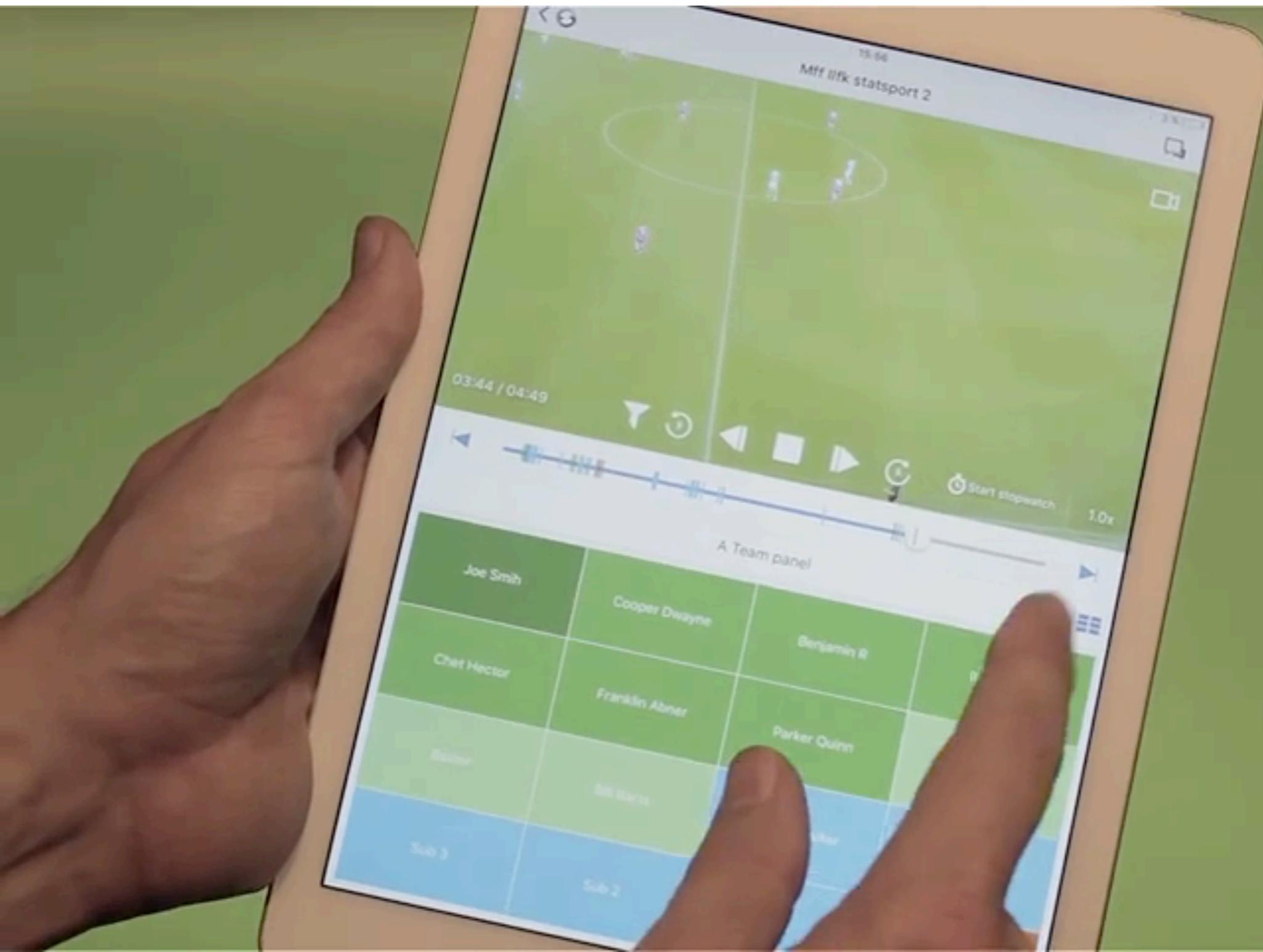
- Machine Learning
 - Classification
 - Bayes Rule
 - Estimating density functions
 - Counting
 - Binning
 - Adaptive binning (k-means)
 - Parametric density estimation
 - Plug in estimated densities
 - **Plug-in classifier**
 - **NN and K-NN**
 - **Logistic Regression**
 - Clustering
 - **K-means**

Master's Thesis Suggestion of the day

- Make a system that finds, detects and tracks football players



Image Analysis - Motivation





LUND
UNIVERSITY

350