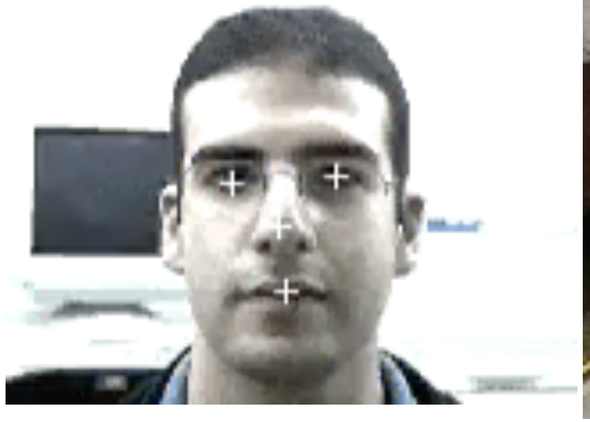
LECTURE 11: INTRODUCTION TO BAYESIAN LEARNING AND THE NAÏVE BAYESIAN CLASSIFIER

Volker Krüger

LUND UNIVERSITY

WHO AM I?



LUND UNIVERSITY

- K-mea
- K-Nea
- Classi

- What
  - we
  - we
  - unc
  - usin

IMAGENET Large Scale Visual Recognition Challenge

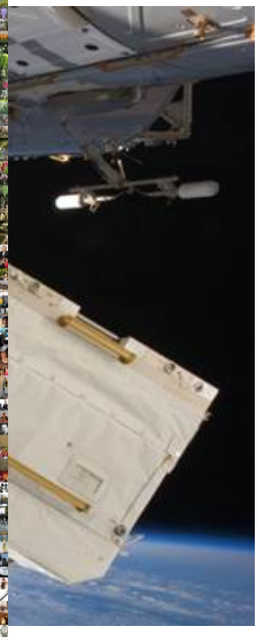
The Image Classification Challenge:
1,000 object classes
1,431,167 images

Output:
Scale
T-shirt
Steel drum
Drumstick
Mud turtle

✔

Output:
Scale
T-shirt
Giant panda
Drumstick
Mud turtle

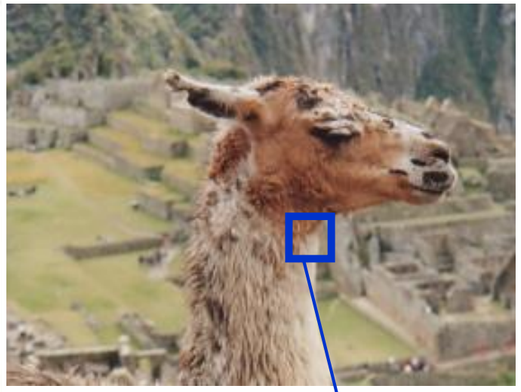
✗

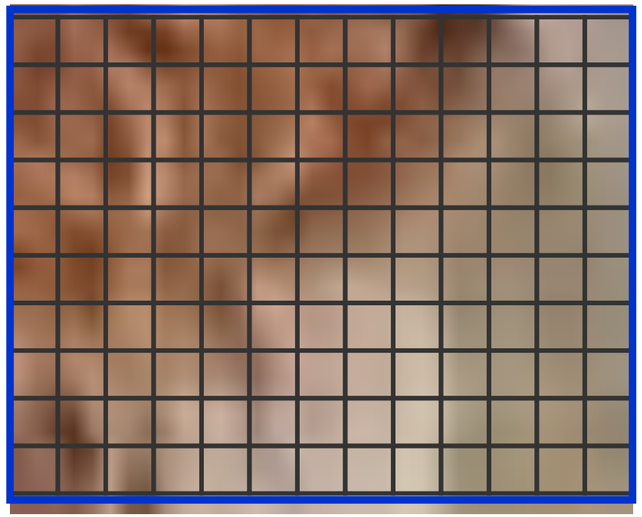


Russakovsky et al. IJCV 2015

LUND UNIVERSITY

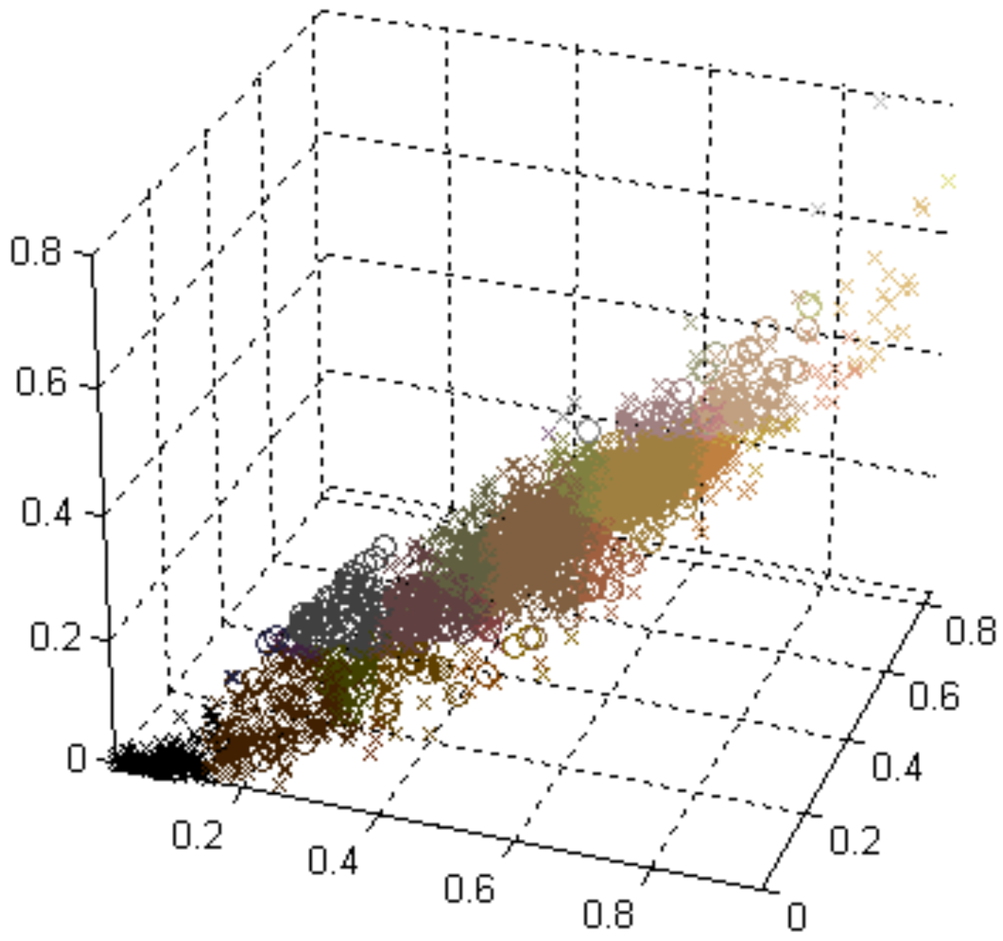
EXAMPLES: SEGMENTATION IN CAMOUFLAGE

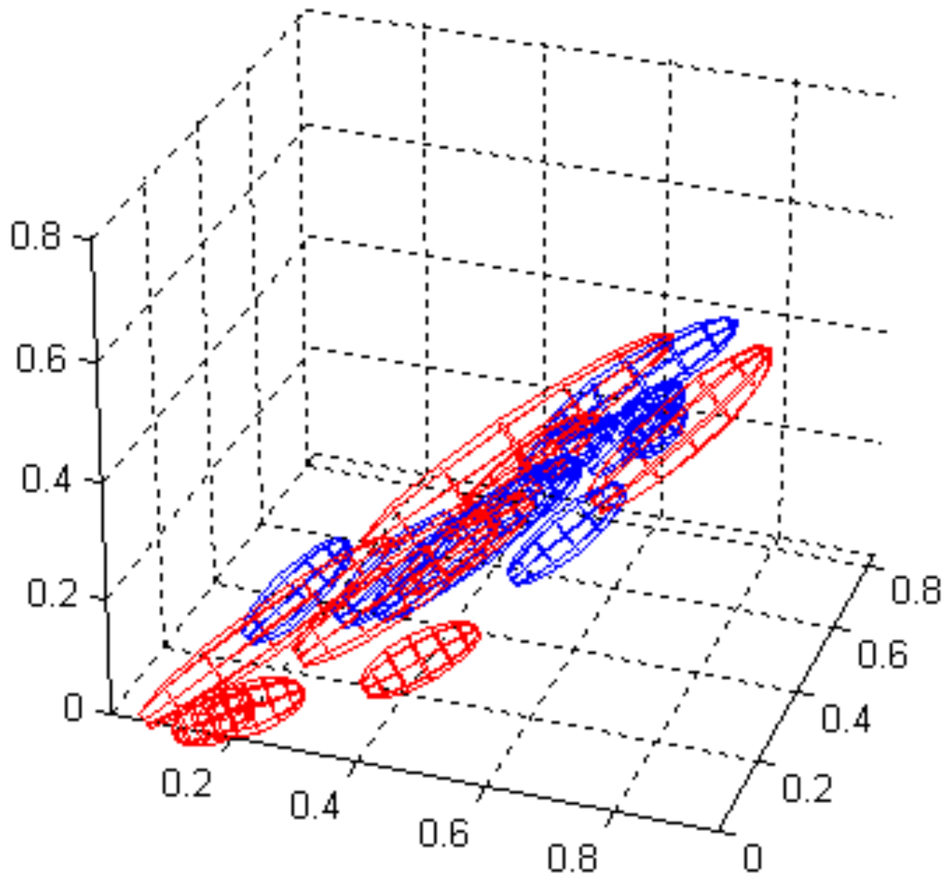




coherent foreground

NEED FOR SUITABLE STATISTICAL APPROACH

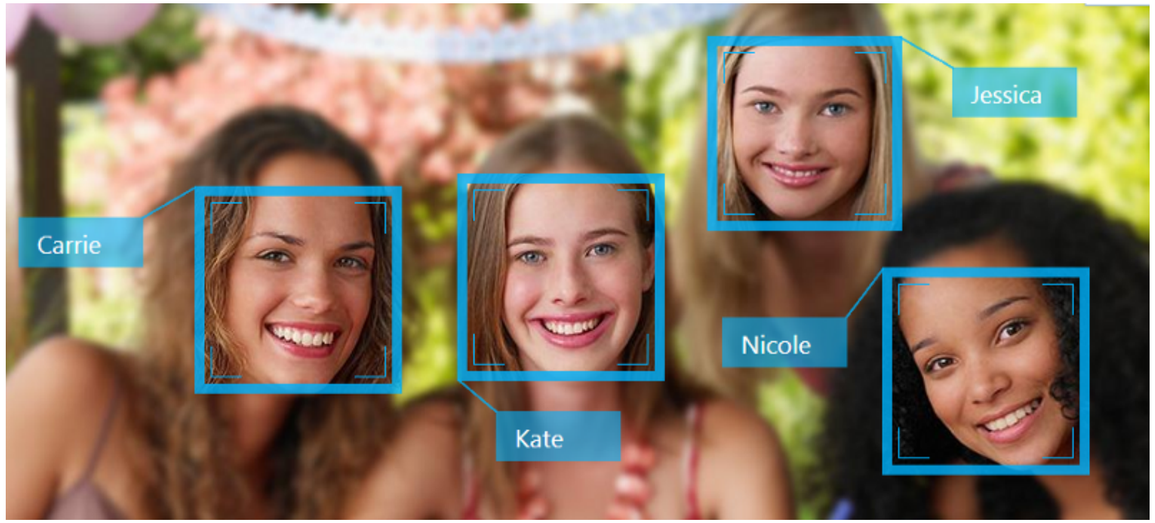




LUND UNIVERSITY

MORE EXAMPLES

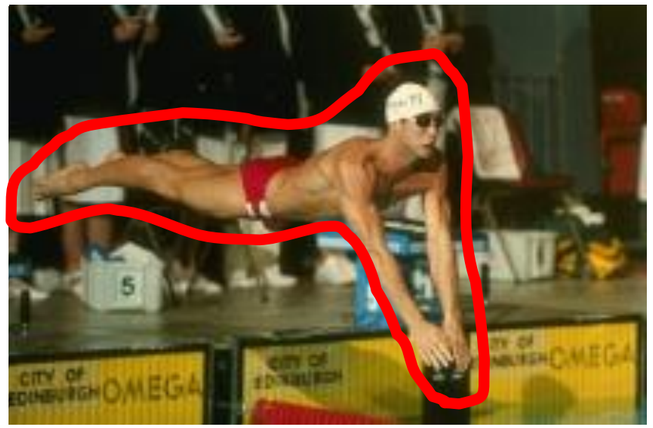
Face recognition: very few examples of an individual



Easy face recognition in openCV

Difficult structures

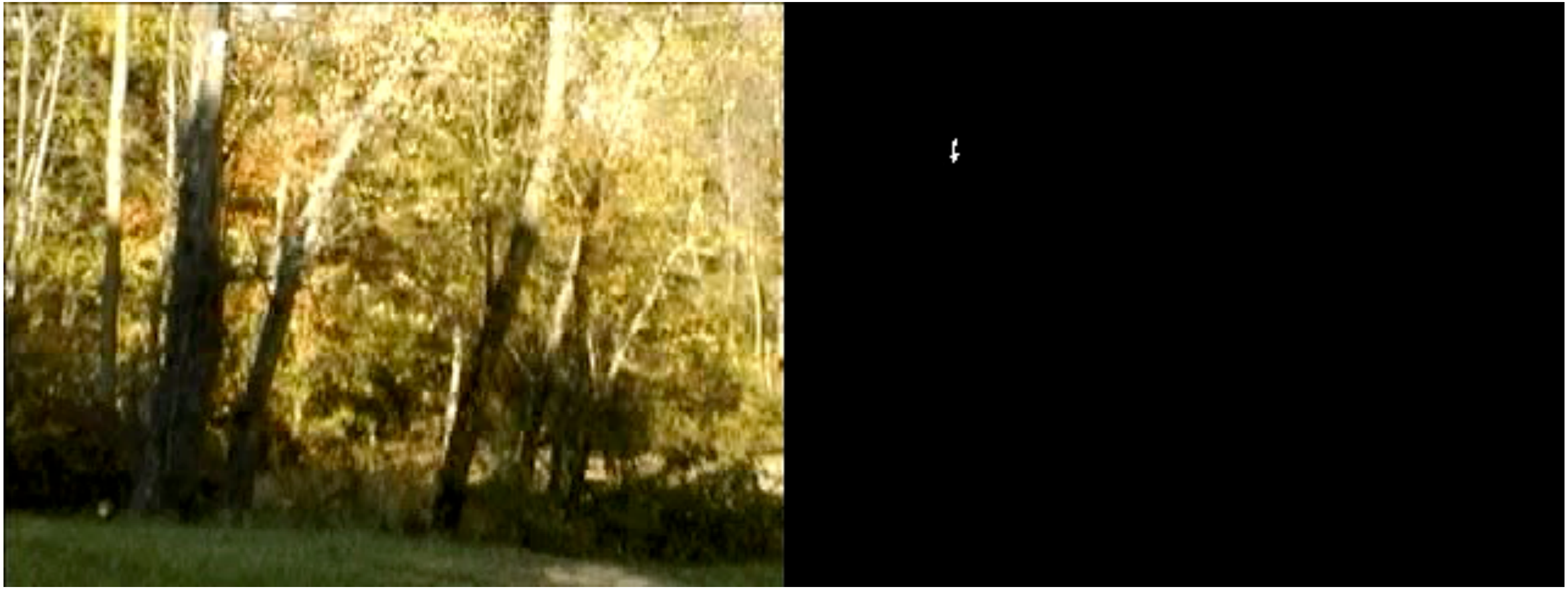
No telepathy



LUND UNIVERSITY

LUND UNIVERSITY

BACKGROUND SUBTRACTION



LUND UNIVERSITY

PLAN FOR TODAY

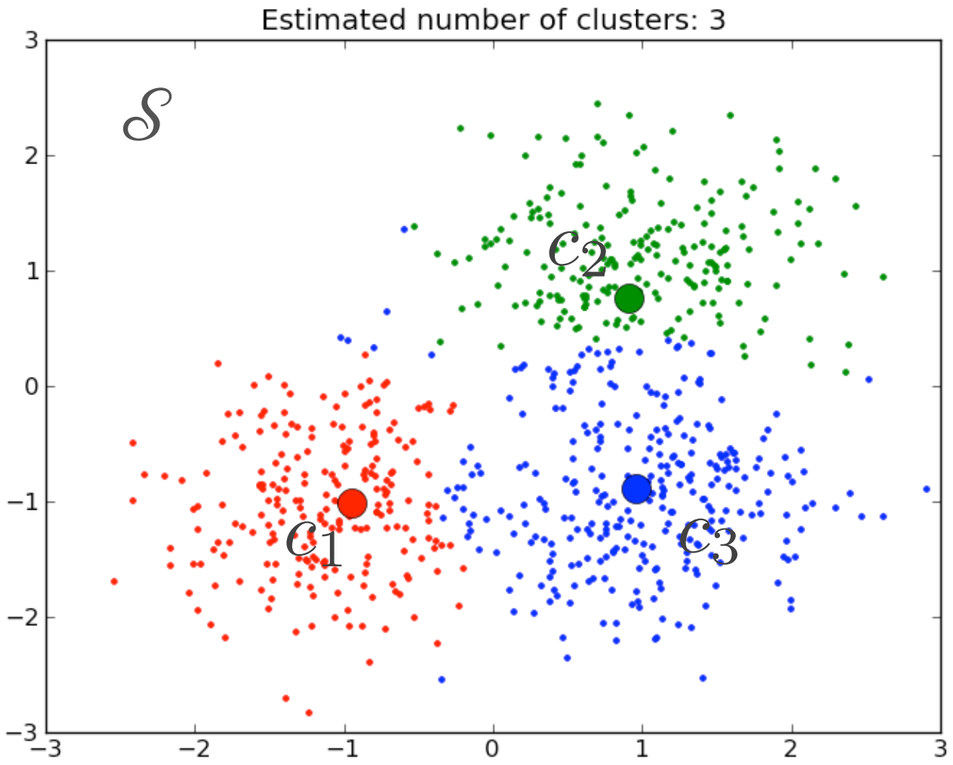
- In the area of supervised learning
- Introduce the Nearest Centroid Classifier (NCC).
- Bayesian formulation of likelihood measurement
- Naïve Bayesian Classifier
  - Continuous
  - Discret

NEAREST CENTROID CLASSIFIER

- Compute class-specific single feature vectors:
  - Compute the mean feature vector from all exemplars of each single class $\mathcal{C}_i$.

$$c_i = \frac{1}{n_i} \sum_{j=1}^{n_i} x_j^i$$

- For estimating the object class of a new sample $x \in \mathcal{S}$, we find the nearest centroid.



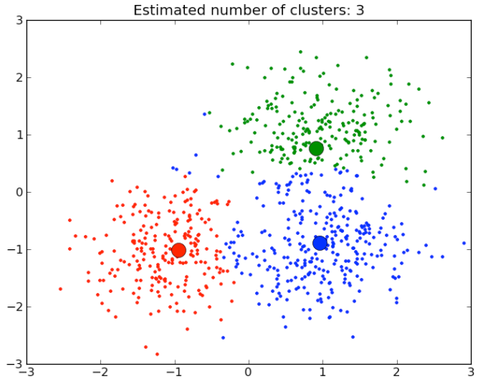
$$x \in \mathcal{C}_i \iff i = \arg\min_j \|x - c_j\| \quad \text{in case of} \quad x, c_i \in \mathbb{R}^m$$

LUND UNIVERSITY

PROBABILISTIC REFORMULATION

- $x \in \mathcal{S}$ is correctly called a ***sample***, often the intuitive terms *exemplar* or *example* are used.
- In general, $c_i = \frac{1}{n} \sum_{i=1}^{n} x_i$ is called **sample mean** of the samples $\{x_1, x_2, \ldots, x_n\}$
- c_i are called ***sample means of class*** i

PROBABILISTIC REFORMULATION



- Let's assume we have a number of samples given for different classes $\mathcal{C}_k$

class label

$$\mathcal{C}_k = \{x_1^k, x_2^k, \ldots, x_{n_k}^k\}$$

- Based on the class samples we can compute the class centroids: $\mathcal{R} = \{c_1, c_2 \ldots, c_n\}$

$$c_k = \frac{1}{n} \sum_{i=1}^{n} x_i$$

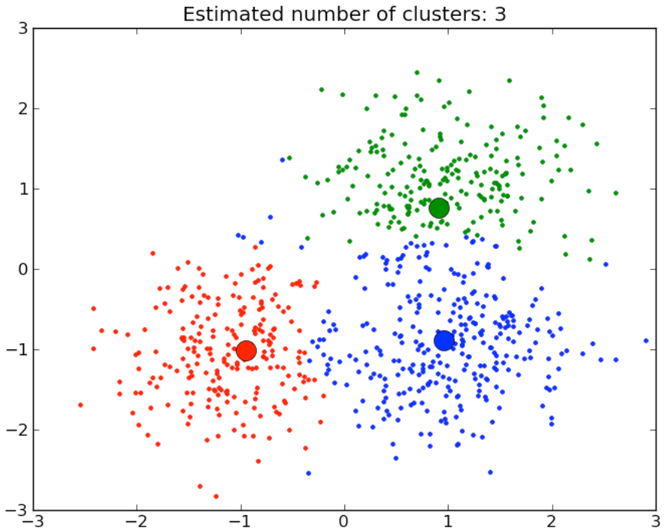
- To test a new sample $x \in \mathcal{S}$

- Probabilistically formulated, we say we find the c_i that has most likely the same class as x

$$i = \text{argmax}_j \mathcal{P}(x|y = j)$$

CONDITIONAL PROBABILITIES

$$\mathcal{P}(x|y=i)$$

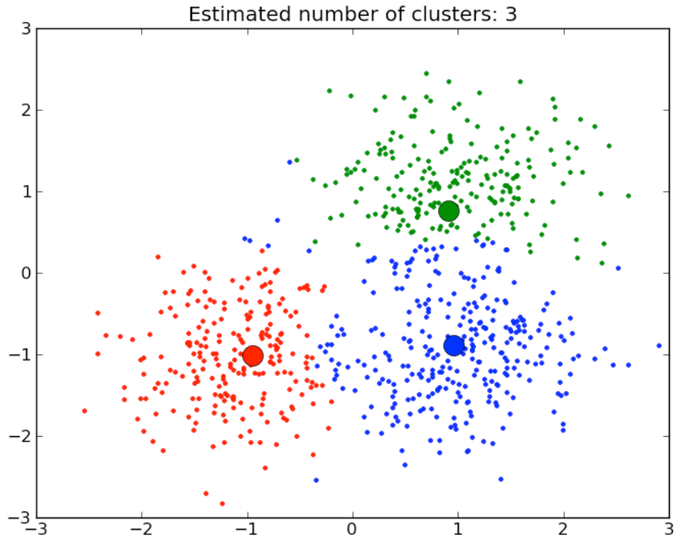


- you can interpret this also as
  - "assuming *that the class y=i*, how likely is *the observation x part of that class*?"
  - "given the model for *y=i*, how good does *x* fit that model?"
  - "how well does the model *y=i* explain the sample point *x*?"

PROBABILISTIC REFORMULATION

For NCC: For a given c_i what is the *likelihood* that c_i and x have the same class?

$$\mathcal{P}(x|y=i)$$



MAXIMUM LIKELIHOOD FORMULATION

Find the class $y = i$ that *maximises* the *likelihood* that x belongs to class i

$$\text{maximize } \mathcal{P}(x|y = i) \text{ over all classes } i$$

PROBABILISTIC FORMULATION

- this formulation is a *purely semantic* formulation: all we are saying is that we are asking now for the ***likelihood*** $\mathcal{P}(x|y=i)$
- We have not yet provided a formula, a distance measure or anything:
  - we have not yet said *how* we want to compute the likelihood.
- This is exactly, what we need to do next: We must specify how we want to compute

$$\mathcal{P}(x|y=i)$$

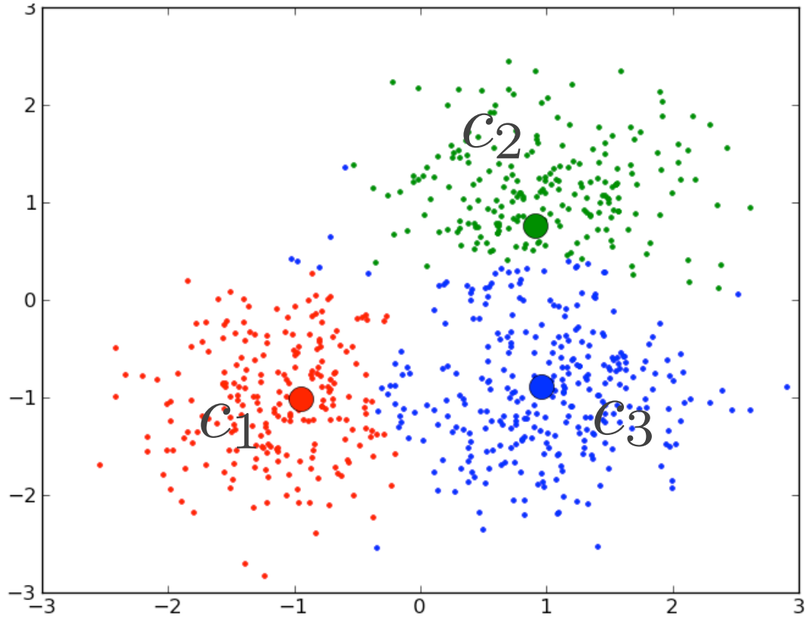
PROBABILISTIC FORMULATION

- Note:
  - $\mathcal{P}(x|y)$ denotes first of all a formula with two arguments: x and the class y
  - We must specify, which argument is fix and over which argument we maximize.
  - In our case,
    - x is fix, because that is the observation of which we want to know the class.
    - c_i is the *model* of each class $y = i$. Thus, we are looking for the model that fits best to the observation
  - We also sometime say: *we are looking for the model that explains the observation best.*

NEAREST CENTROID CLASSIFIER

Model for each class $y = i$: $c_i = \frac{1}{n_i} \sum_{j=1}^{n_i} x_j^i$ summing over all class members.

The c_i are the ***models*** of the classes.



$$x \in \mathcal{C}_i \iff i = \arg\min_j \|x - c_j\| \quad \text{in case of} \quad x, c_i \in \mathbb{R}^m$$

LUND UNIVERSITY

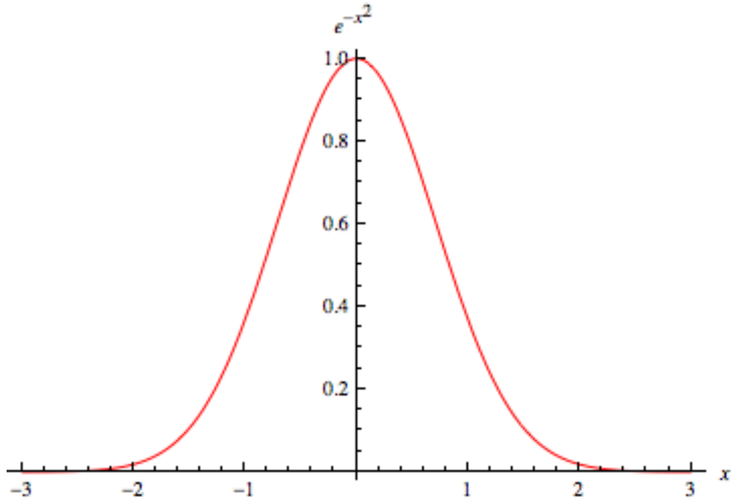
NEAREST CENTROID CLASSIFIER

So far: $\|c_i - x\|$

Values are within $[0, \infty)$

Now: $\mathcal{P}(x|y=i) = \frac{1}{\sqrt{2\pi}} \exp^{-\frac{1}{2}(c_i - x)^2}$

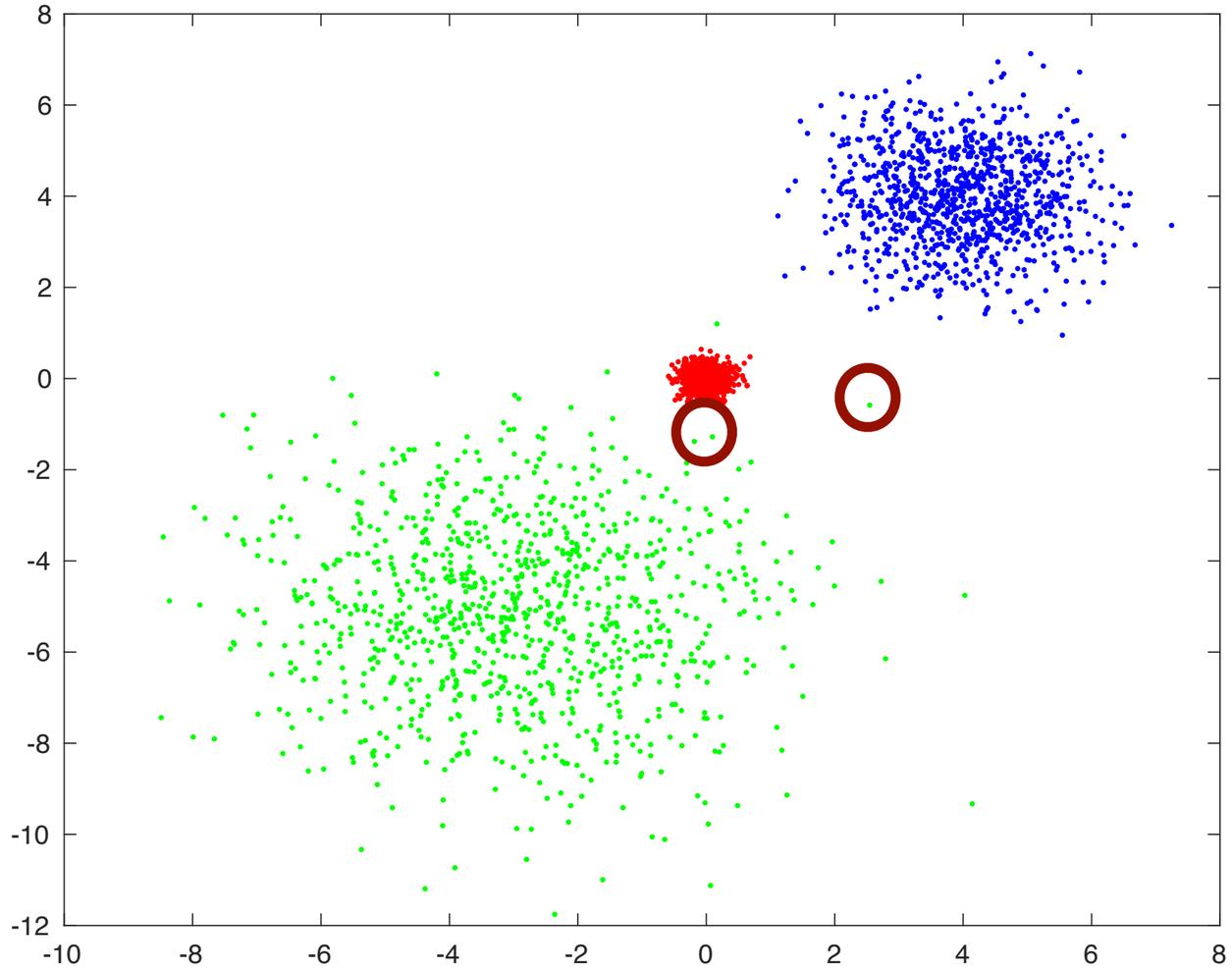
with values within $(0, 1]$



Gaussian function

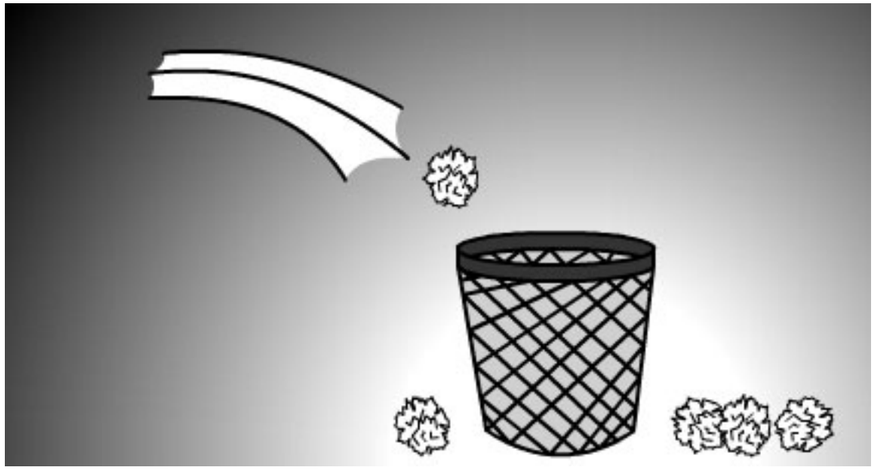
EXAMPLE: WHAT HAPPENS IF…?

- What happens if the distribution are very different?
- Is finding the closest mean sufficient?



GAUSSIAN DISTRIBUTION

- **Observation:**
  - Points of each class are distributed around a single common ***mean*** of that class.
  - Different classes may have different ***deviations*** around their mean.
- When trying to identify a class, we should take into consideration this deviation!!!
  - *uncertainty* is an intuitive term.
- In correct terminology, the uncertainty is given by the ***covariance*** of the samples.
- Note: in 1D we use the terms ***variance*** and ***standard deviation.***



1D-GAUSSIAN DISTRIBUTION

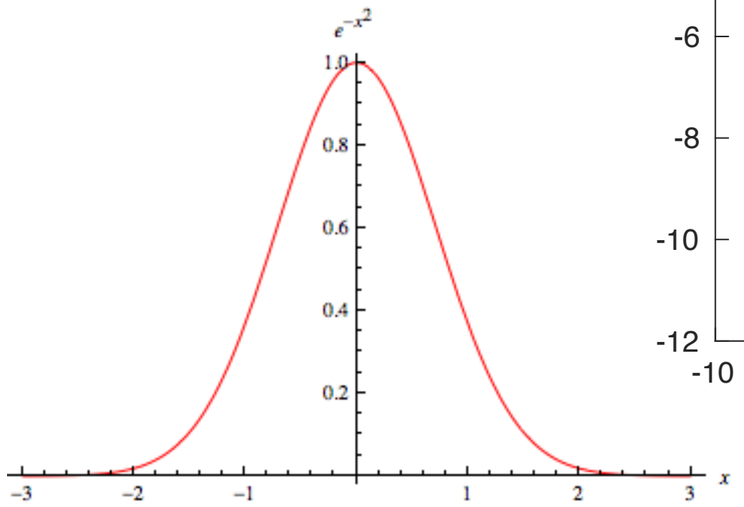
The 1D Gaussian is *completely* defined through its mean and variance:

Mean $$E[X] = \mu = \frac{1}{n}\sum_{i=1}^{n} x_j \quad x_j \in \mathbb{R}$$

Variance $$\mathrm{Var}(X) = \sigma_X^2 = E\left[(X - E[X])^2\right] = E\left[(X-\mu)^2\right]$$
$$= \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2$$

Standard deviation σ_X

$$\mathrm{Gauss}(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$



Very small σ^2

Very large σ^2

$$\text{Gauss}(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

kD-GAUSSIAN DISTRIBUTION

The kD Gaussian is completely defined through $E[X]$ and $\mathrm{VAR}(X, Y)$

Mean $E[X] = \mu = (E[x_1], E[x_2], \ldots E[x_k]) \quad x_i \in \mathbb{R}, \quad X \in \mathbb{R}^k$

Covariance

$$COV(X, Y) \quad = \boldsymbol{\Sigma} \quad = E\left[(X - E[X])(Y - E[Y])\right]$$

$$COV(x_i, x_j) \quad = \boldsymbol{\Sigma}_{\mathbf{i},\mathbf{j}} \quad = E\left[(x_i - E[x_i])(x_j - E[x_j])\right]$$

$$= \frac{1}{n-1} \sum_{k=1}^{n} (x_i^k - \mu_i)(x_j^k - \mu_j)$$

$$\mathrm{Gauss}(\mathbf{X}, \mu, \boldsymbol{\Sigma}) = (2\pi)^{-k/2} |\boldsymbol{\Sigma}|^{-1/2} e^{-\frac{1}{2}(\mathbf{x}-\mu)' \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\mu)}$$

COMMENTS

- The *N-D Gaussian* looks essentially like the *1-D Gaussian*, but in matrix notation
- The *N-D* mean is computed component-wise like the *1-D* mean
- The covariance matrix takes into considerations statistical dependencies across the different dimensions!
- x_i^k is the k-*th* sample of vector x and we consider *only* the i-*th* component (the i-*th* dimension) of the vector x

$$COV(X,Y) = \frac{1}{n-1}\sum_{k=1}^{n}(x_i^k - \mu_i)(x_j^k - \mu_j)$$

- Averages over the product of the component wise distances of x_i and x_j to their respective means μ_i and μ_j

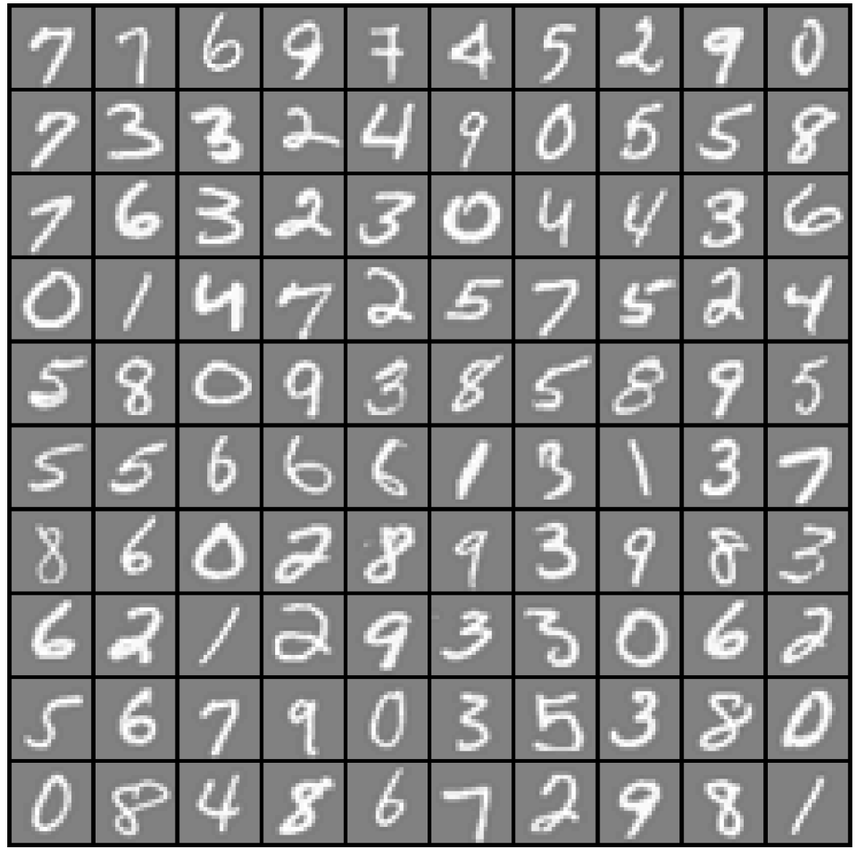
COMMENTS

- Let's assume we had 500 samples per class, and 2-D data.
- How many degrees of freedom does the mean and the covariance have?
- Is 500 samples enough to compute the free parameters?

COMMENTS

- Now, consider object recognition based on images. Image 50x50 pixels
- How many degrees of freedom does the mean and the covariance have?
- How many samples do we need now?

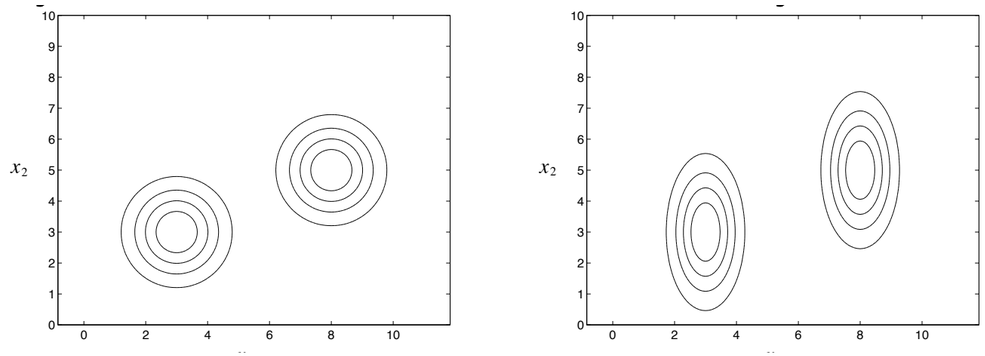
- We have 2500 dofs for the mean and 6250000 for the covariance matrix per class!



This is where CNNs shine in cases where we have enough data!

REGULARISATION IS A SIMPLE TRICK!

- Idea 1: assume all the covariances are the same.



- Idea 2: Make **independence assumptions** to get **diagonal** or **identity-multiple** covariances.
  - here, we assume that, e.g. x_i and x_j are statistically independent:

$$P(x_i, x_j) = P(x_i)P(x_j)$$

- The result is that $\mathrm{cov}_{ij} = 0, \quad i \neq j$!!!!

LUND UNIVERSITY

PROOF

Independence: $P(x,y) = P(x)P(y) \implies E\{xy\} = E\{x\}E\{y\}$

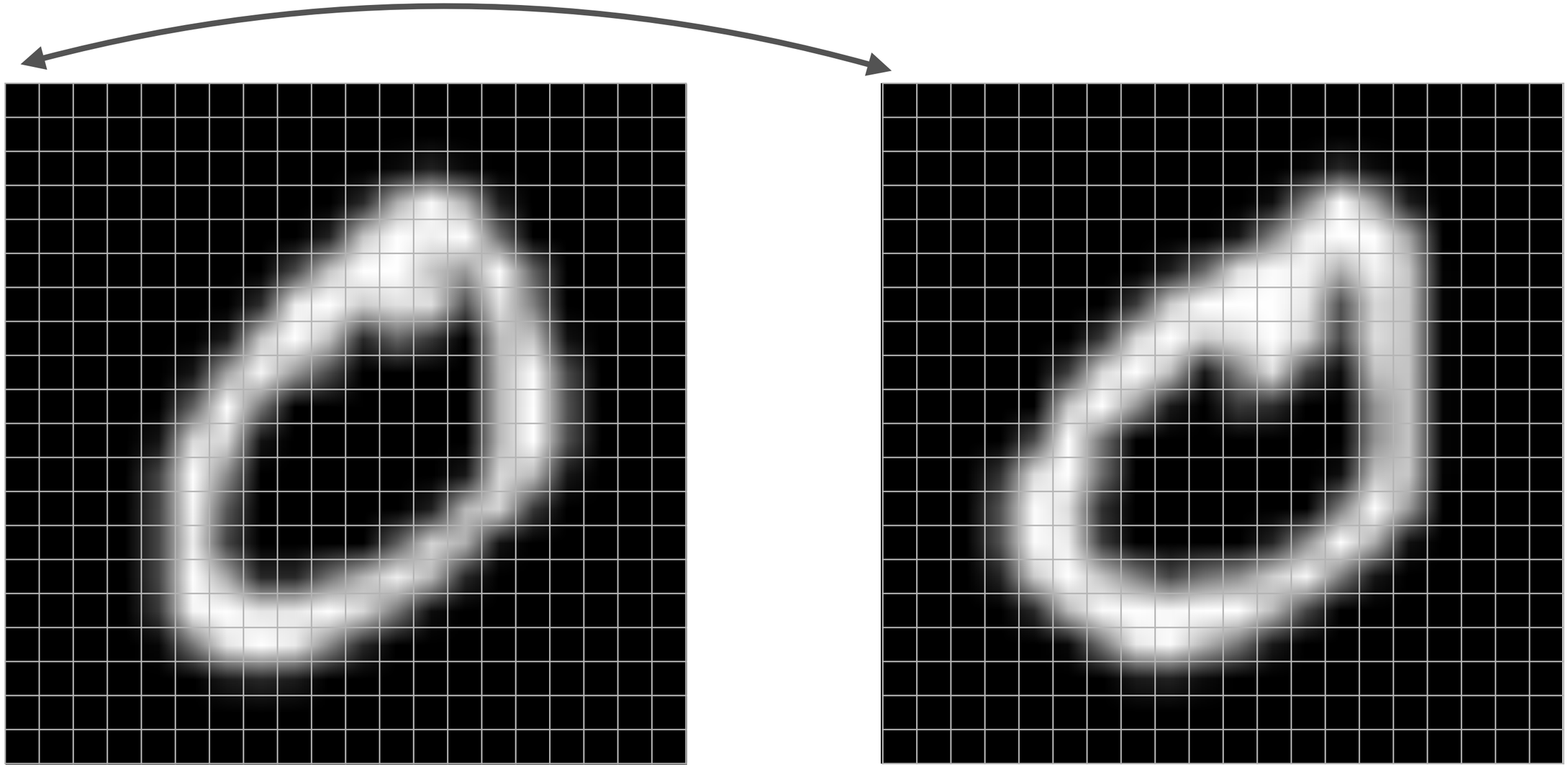
$$
\begin{aligned}
\mathrm{COV}(x,y) &= E\{(x - E\{x\})(y - E\{y\}\} \\
&= E\{xy - E\{x\}y - xE\{y\} + E\{x\}E\{y\}\} \\
&= E\{xy\} - E\{x\}E\{y\} - E\{x\}E\{y\} + E\{x\}E\{y\} \\
&= E\{xy\} - E\{x\}E\{y\} \\
&= 0 \iff x, y \text{ independent}
\end{aligned}
$$

NAÏVE (IDIOT'S) BAYES CLASSIFIER

- The previous *N*-D Gaussian looks very complicated.
- We can use a much simpler version that uses 1D Gaussians for each dimension of the *N*-D space.

$$p(x|y=i) = \prod_i p(x_i|y)$$

- This is called the **Naïve Bayesian Classifier**.
- It assumes that the covariance matrix is diagonal!
  - it assumes *independency* between the different dimensions.
- It works surprisingly well!



$$p(x|y=i) = \prod_i p(x_i|y)$$

ML, MAP AND LOG-PROBABILITY

Classification rules:

- Maximum Likelihood (ML): $\mathrm{argmax}_i \mathcal{P}(x|y=i)$
- Maximum A-posteriori (MAP): $\mathrm{argmax}_i \mathcal{P}(y=i|x) = \mathrm{argmax}_i \mathcal{P}(x|y=i)\mathcal{P}(y=i)$

posterior probability

Prior

Using the log makes the equations simpler!

$$\log \mathcal{P}(y=i|x) = \log \mathcal{P}(x|y=i) + \log \mathcal{P}(y=i)$$

MAP is often called *penalised ML*.

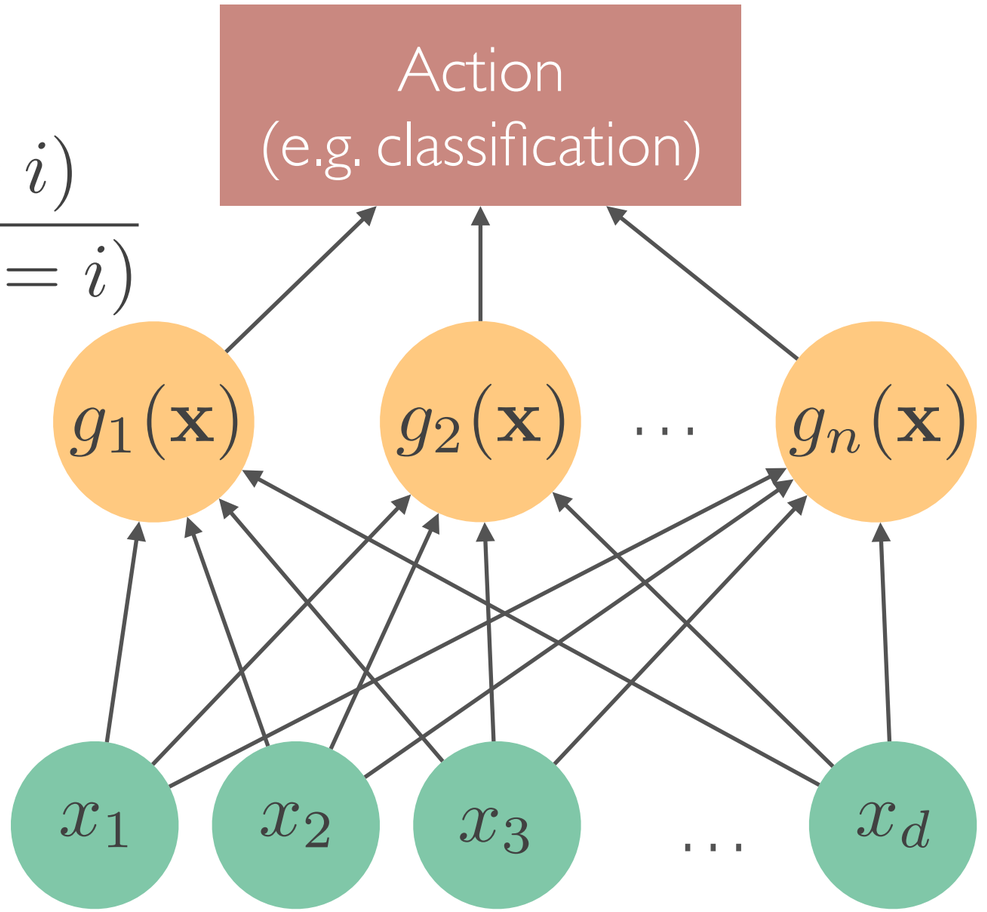
BAYESIAN CLASSIFIER

$$g_i(\mathbf{x}) = \mathcal{P}(y = i, \mathbf{x}) = \frac{p(\mathbf{x}|y = i)\mathcal{P}(y = i)}{\sum_i p(\mathbf{x}|y = i)\mathcal{P}(y = i)}$$

$$g_i(\mathbf{x}) = p(\mathbf{x}|y = i)\mathcal{P}(y = i)$$

$$g_i(\mathbf{x}) = \ln p(\mathbf{x}|y = i) + \ln \mathcal{P}(y = i)$$

are all equivalent



DISCRETE BAYESIAN CLASSIFIER

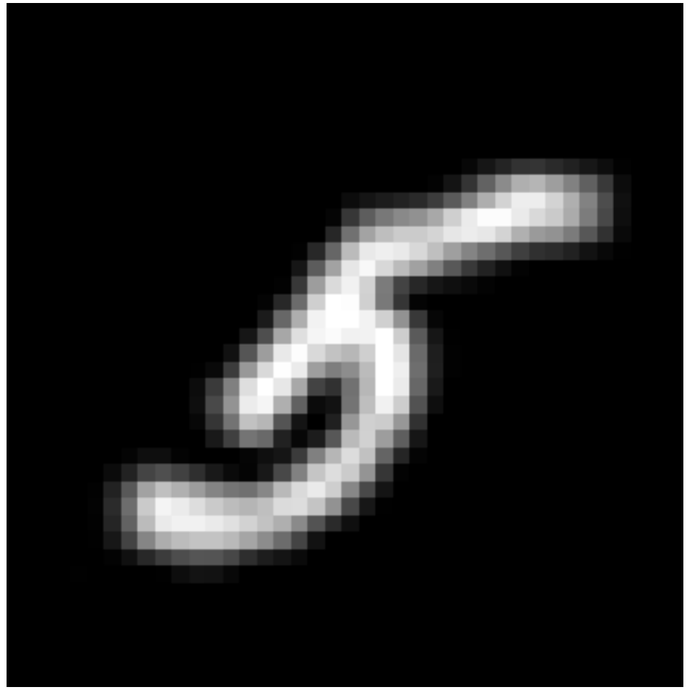
- Before the break, we have dealt with continues data from $\mathbb{R}^n$
- However, what to do if the data is categorial or discrete?

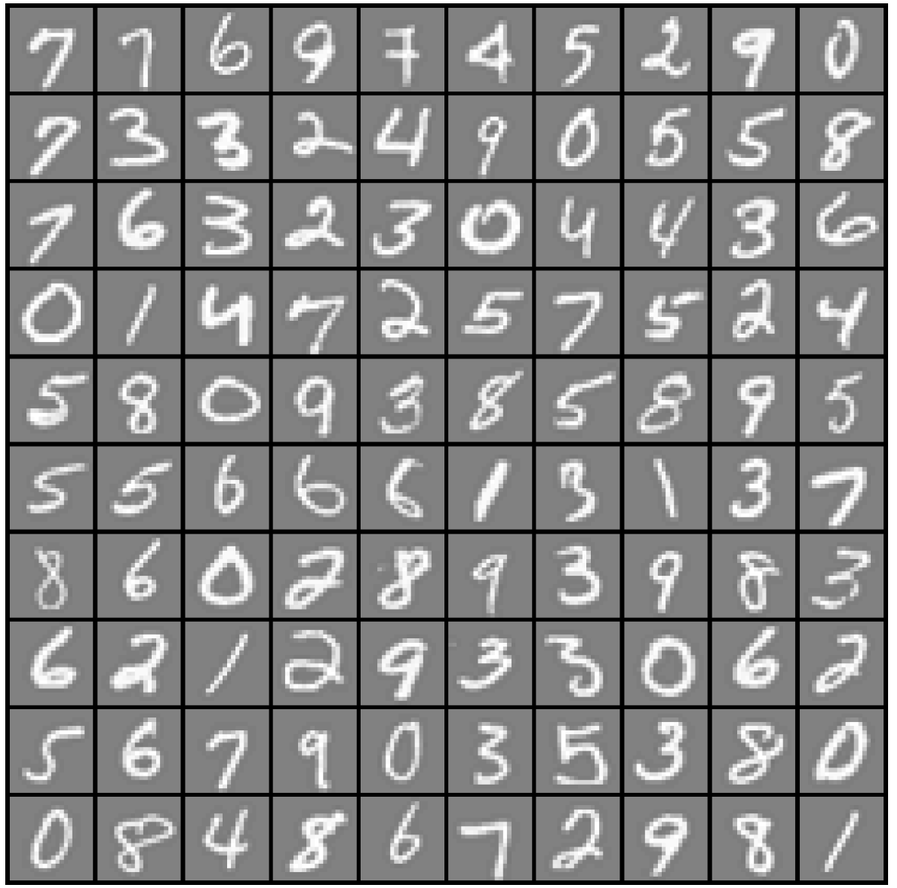
EXAMPLE OF DISCRETE DATA

- Consider an automated phone answering system where customers are supposed to state their wishes.
- Customer wishes might be concerned with: *bill, techSupport, operator, appointments, changeService, video, orders, internet, etc.*
- Customer might use words like: *my, support, tech, service, an, to, a, change, help, internet, order, agent, appointment, customer*
- The challenge of the system is to find out from the spoken language what the customer wants and then to connect with the right service.

EXAMPLES OF DISCRETE DATA

- Automated recognition of zip-codes on letters to improve the postal service
- A letter consists of 20x20 pixels, 256 grey values





DISCRETE BAYESIAN CLASSIFIER

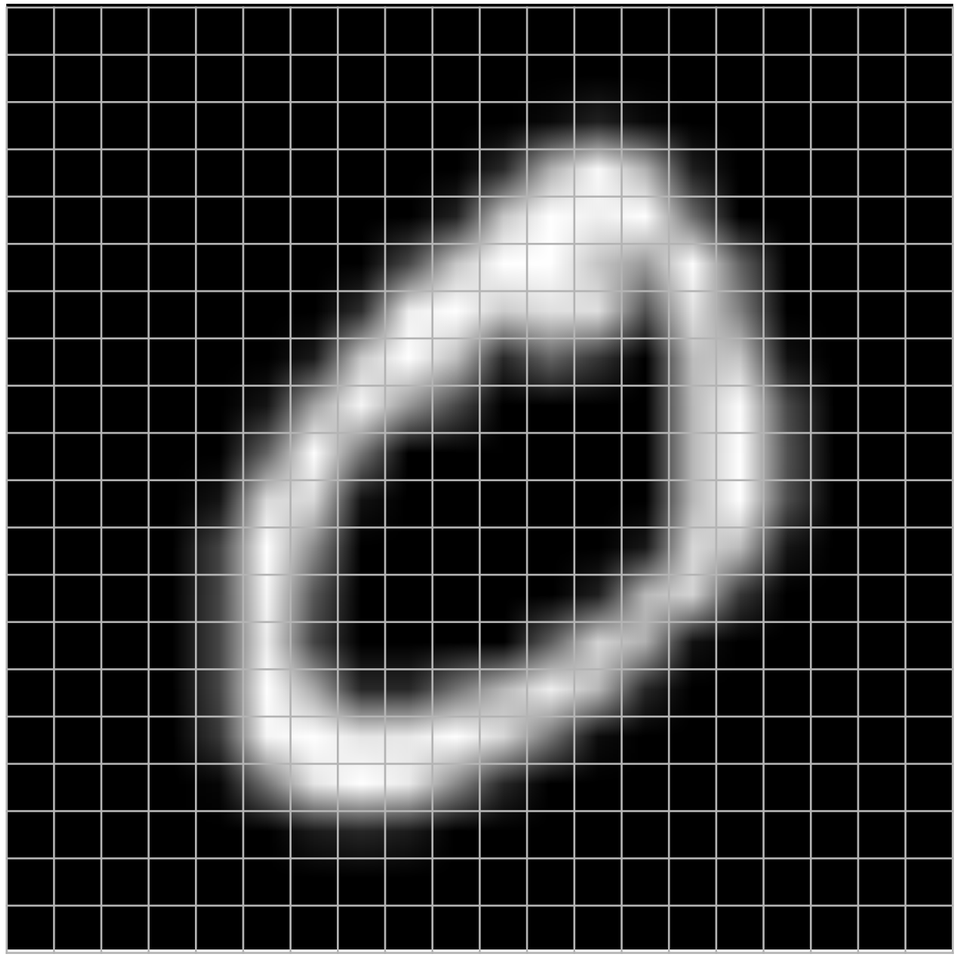
- How about a simple table (joint multinomial) as a class-conditional model?

$$p(x_1 = a, x_2 = b, \ldots | y = c) = \eta^c_{ab\ldots}$$

- Intuitively, count the occurrences:

$$\eta^c_{ab\ldots} = \frac{\sum_n [y_n = c][x_1 = a][x_2 = b][\ldots][\ldots]}{\sum_n [y_n = c]}$$

What is the problem with this?



$$\eta^c_{ab\ldots} = \frac{\sum_n [y_n = c][x_1 = a][x_2 = b][\ldots][\ldots]}{\sum_n [y_n = c]}$$

DISCRETE BAYESIAN CLASSIFIER

- Consider the 20x20 digits at 256 gray levels
- How many entries in the table?
- How many will be zero?

- We need *regularisation*: something that can simplify the problem!
  - how about the independence assumption between the parameters?
  - What is the effect?

DISCRETE NAÏVE GAUSSIAN CLASSIFIER

$$\eta^c_{ab\ldots} = \frac{\sum_m [y^m = c][x_1^m = a][x_2^m = b][\ldots][\ldots]}{\sum_m [y^m = c]}$$

This is averaging!!!

$$\eta^*_{ijk} = \frac{\sum_m [x_i^m = j][y^m = k]}{\sum_m [y^m = k]}$$

Discrete features x_i, assumed independent given the class label y.

how often does $x_i = j$ *appear in class* k

$$p(x_i = j | y = k) = \eta_{ijk}$$

$$p(x_1 = j_1, x_2 = j_2 \ldots | y = k, \eta) = \prod_i p(x_i = j_i | y = k, \eta) = \prod_i \prod_j \eta_{ijk}^{[x_i = j]}$$

Wait a second!!
Is this really how we should do it?

We followed our intuition and said: We just count the occurrences of the words / pixels and this way essentially compute the average. But is this intuition actually correct?

$$\eta^*_{ijk} = \frac{\sum_m [x_i^m = j][y^m = k]}{\sum_m [y^m = k]}$$

What is the problem really?

We have a whole lot of observed data, and we need to estimate the model θ that has most likely generated the data.

EXAMPLE

Throwing a dice, we expect roughly the same occurrences of each number.

Why?

Because the model θ says that each side appears with a likelihood of $\frac{1}{6}$

And how can we compute θ from observations?

$$p(x = 6) = \frac{\#\text{6es}}{\#\text{throws}}$$

MAXIMUM LIKELIHOOD ESTIMATION

Likelihood estimation: $p(x|\theta)$ *What is the likelihood of* x *given the model* θ

Lets define the Log-Likelihood estimation: $l(x, \theta) = \log p(x|\theta)$

Note that $l(x, \theta)$ is a 2D function where we might know θ and want to estimate the $p(x|\theta)$

EXAMPLE

For our Gaussians from the beginning, θ would contain the means and the Covariances of the Gaussians. $\theta = (c_1, \Sigma_1, c_2, \Sigma_2, c_3, \Sigma_3)$

EXAMPLE

Note that $l(x, \theta)$ is a 2D function where we might know $x_1, x_2, \ldots x_m$ and want to know $p(x|\theta)$

Then, we just find the θ that maximises $p(x|\theta)$

$$\theta = \arg\max_{\theta} l(\theta, x_1, x_2, \ldots, x_m)$$

This is exactly(!!) what we did before: Maximum Likelihood Estimation

INDEPENDENTLY, IDENTICALLY DISTRIBUTED

What is the problem here? $\theta = \arg\max_{\theta} l(\theta, x_1, x_2, \ldots, x_m)$

Order might matter!!

Imagine x_i being weather observation where x_{i+1} depends on x_i.
Here, x_{i+1} is not only statistically dependent on x_i, its distribution is might also be completely different depending on the value of x_{i+1}.

But there are cases, where the order does *not* matter and all samples are from the same distribution! The dice is such an example. This is called

Independently and Identically Distributed (IID)

INDEPENDENTLY AND IDENTICALLY DISTRIBUTED

So what is the point of data being IID?

1. The ordered observations $x_1, x_2, \ldots x_m$ can be just put into an unordered set $\mathcal{D} = \{x_1, x_2, \ldots x_m\}$

2. We can make use of the statistical independence: $\mathcal{P}(X, Y) = \mathcal{P}(X)\mathcal{P}(Y)$

$$p(\mathcal{D}|\theta) = \prod_{i=1}^{m} p(x_i|\theta)$$

MAXIMUM LIKELIHOOD OF θ

In our case of learning θ

$$l(\theta, \mathcal{D}) = \sum_{i=1}^{m} \log p(x_i|\theta)$$

$$\theta^* = \text{argmax}_\theta l(\theta, \mathcal{D})$$

EXAMPLE WITH A COIN

- We observe M coin flips: $\mathcal{D} = \mathrm{H},\mathrm{H},\mathrm{T},\mathrm{H},\ldots$
- Model: $p(H) = \theta \;\; p(T) = (1-\theta)$

- Likelihood:

$$\begin{aligned}\ell(\theta;\mathcal{D}) &= \log p(\mathcal{D}|\theta) \\ &= \log \prod_m \theta^{\mathbf{x}^m} (1-\theta)^{1-\mathbf{x}^m} \\ &= \log\theta \sum_m \mathbf{x}^m + \log(1-\theta)\sum_m (1-\mathbf{x}^m) \\ &= \log\theta N_{\mathrm{H}} + \log(1-\theta) N_{\mathrm{T}}\end{aligned}$$

- Take derivatives and set to zero:

$$\frac{\partial \ell}{\partial \theta} = \frac{N_{\mathrm{H}}}{\theta} - \frac{N_{\mathrm{T}}}{1-\theta}$$

$$\Rightarrow \theta^*_{\mathrm{ML}} = \frac{N_{\mathrm{H}}}{N_{\mathrm{H}} + N_{\mathrm{T}}}$$

What about our Gaussian?

UNIVARIATE NORMAL DISTRIBUTION

- We observe M iid real samples: $\mathcal{D} = .18, -1.15, .83$ ...
- Model: $p(x) = (2\pi^2)^{-1/2} \exp\{-(x-\mu)/2\sigma^2\}$
- Likelihood (using probability density):

$$\begin{aligned}\ell(\theta; \mathcal{D}) &= \log p(\mathcal{D}|\theta) \\ &= -\frac{M}{2}\log(2\pi\sigma^2) - \frac{1}{2}\sum_m \frac{(x^m - \mu)^2}{\sigma^2}\end{aligned}$$

- Take derivatives and set to zero:

$$\begin{aligned}\frac{\partial \ell}{\partial \mu} &= (1/\sigma^2)\textstyle\sum_m (x_m - \mu) \\ \frac{\partial \ell}{\partial \sigma^2} &= -\frac{M}{2\sigma^2} + \frac{1}{2\sigma^4}\textstyle\sum_m (x_m - \mu)^2 \\ \Rightarrow \mu_{\mathrm{ML}} &= (1/M)\textstyle\sum_m x_m \\ \sigma^2_{\mathrm{ML}} &= (1/M)\displaystyle\sum_m (x_m - \mu_{\mathrm{ML}})^2\end{aligned}$$

FINAL REMARKS

- Maximum Likelihood estimation of the parameters is by far not the best way to estimate parameters, but it is a very good base line.
- It is important to verify if the data is IID!

WHAT WE HAVE LEARNED TODAY

- Intuitive Nearest Centroid Classifier
- Bayesian Formulation of it
  - Likelihoods, Maximum likelihood formulation
- Naïve Bayesian Classifier for continuous data.
- we discussed ML vs MAP by introducing priors
- we investigated the Bayesian classifier for discrete data
- We discussed that IID is a VERY important concept

EXERCISE

1. Write an Nearest Centroid classifier (NCC) in python using the Euclidian distance as in slide 9.
2. Program the NCC using the naïve Bayesian classifier, and compare the results.
   - to take into consideration the uncertainty or standard deviation use the function "cov" from numpy to compute the covariance given the data.
   - How does the recognition rate improve?