Bayesian Classifiers / unsupervised learning

Applied Machine Learning (EDAN96)
Lecture 13
2023-12-11
Elin A. Topp

Material based on Lecture Slides on Probabilistic Representation and Bayesian Learning, EDAF70, Spring 2018,
Lecture 11, EDAN95 Fall 2018
Goodfellow et al, "Deep Learning", and Russel/Norvig, "AI - A Modern Approach"

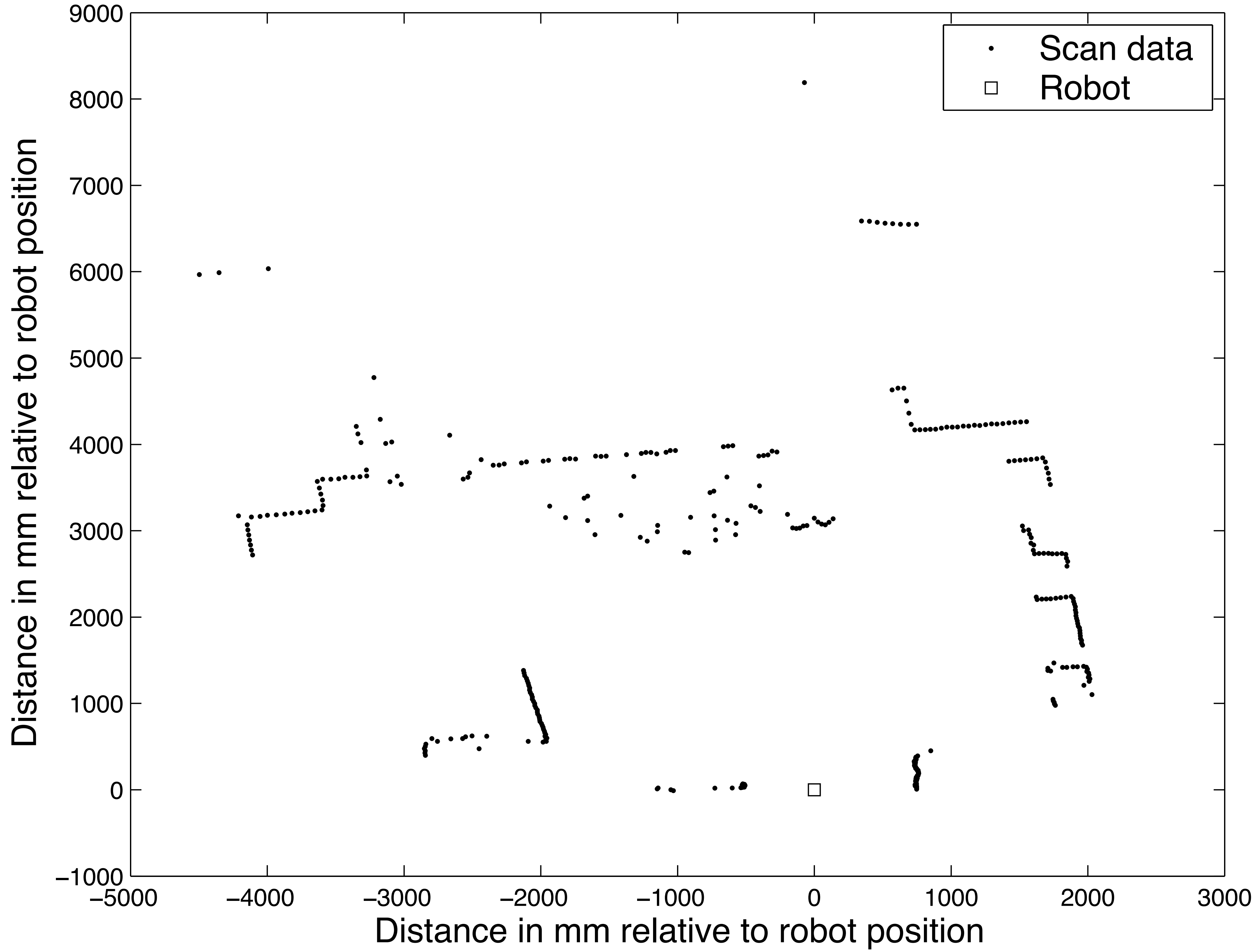
Today's agenda

- Recap (if necessary) conditional / posterior probabilities, Bayes' rule, independence / conditional independence
- The Naive Bayesian Classifier and the Gaussian Naive Bayesian Classifier
- Learning a Bayesian Classifier
- The unsupervised version: EM (for GMMs or k-Means)

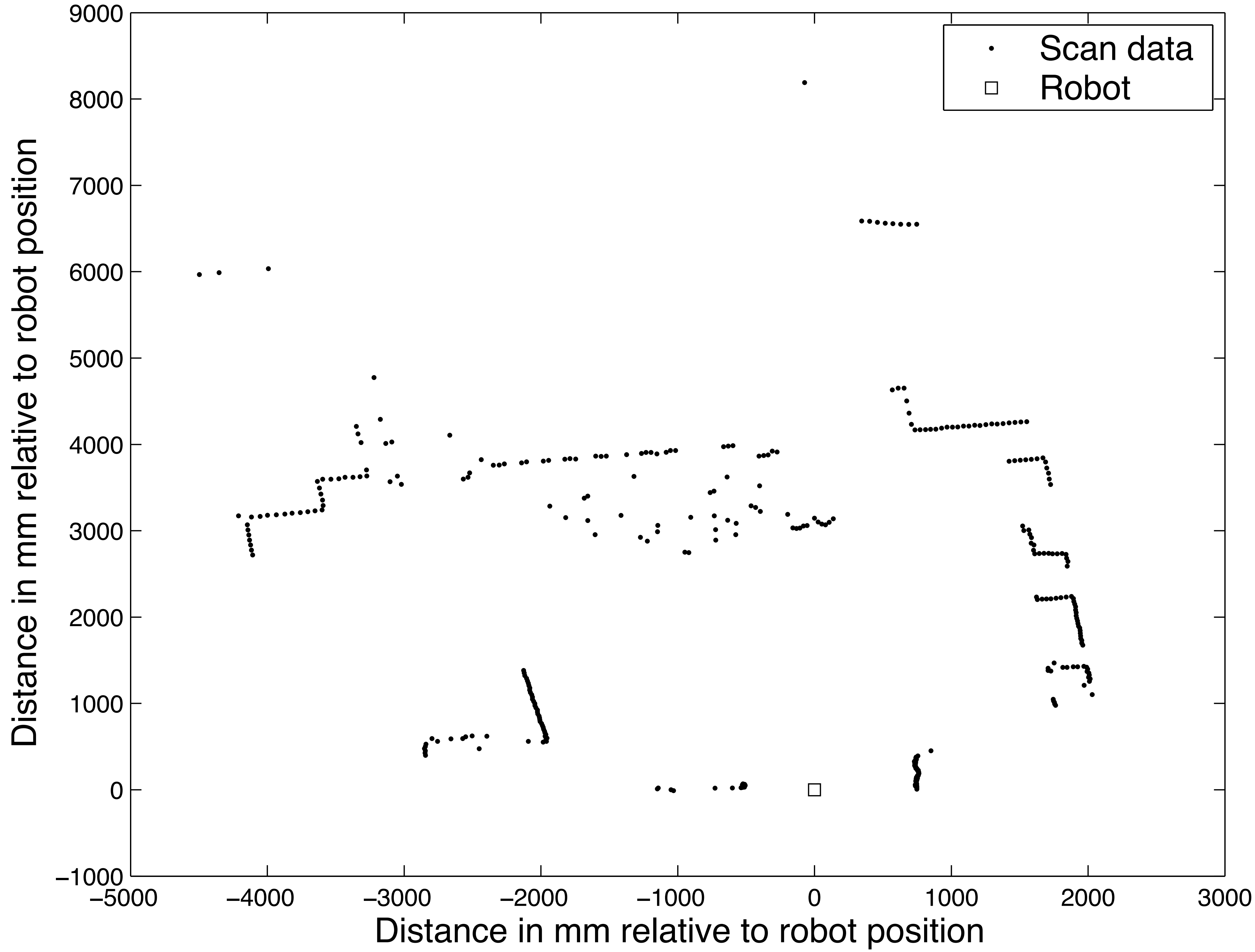
Today's agenda

- Recap conditional / posterior probabilities, Bayes' rule, independence / conditional independence
- The Naive Bayesian Classifier and the Gaussian Naive Bayesian Classifier
- Learning a Bayesian Classifier
- The unsupervised version: EM (for GMMs or k-Means)

A robot's view of the world...

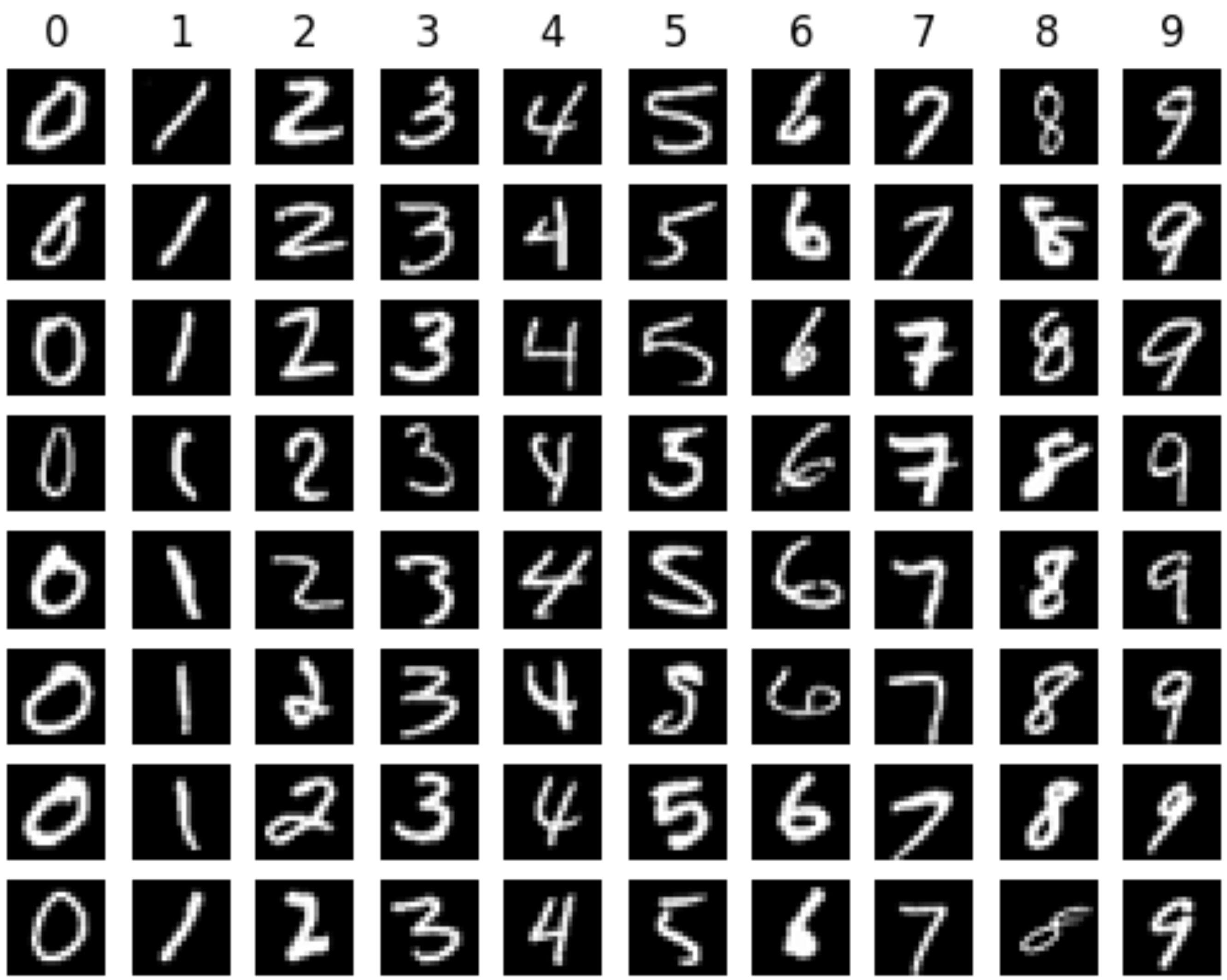


A robot's view of the world...

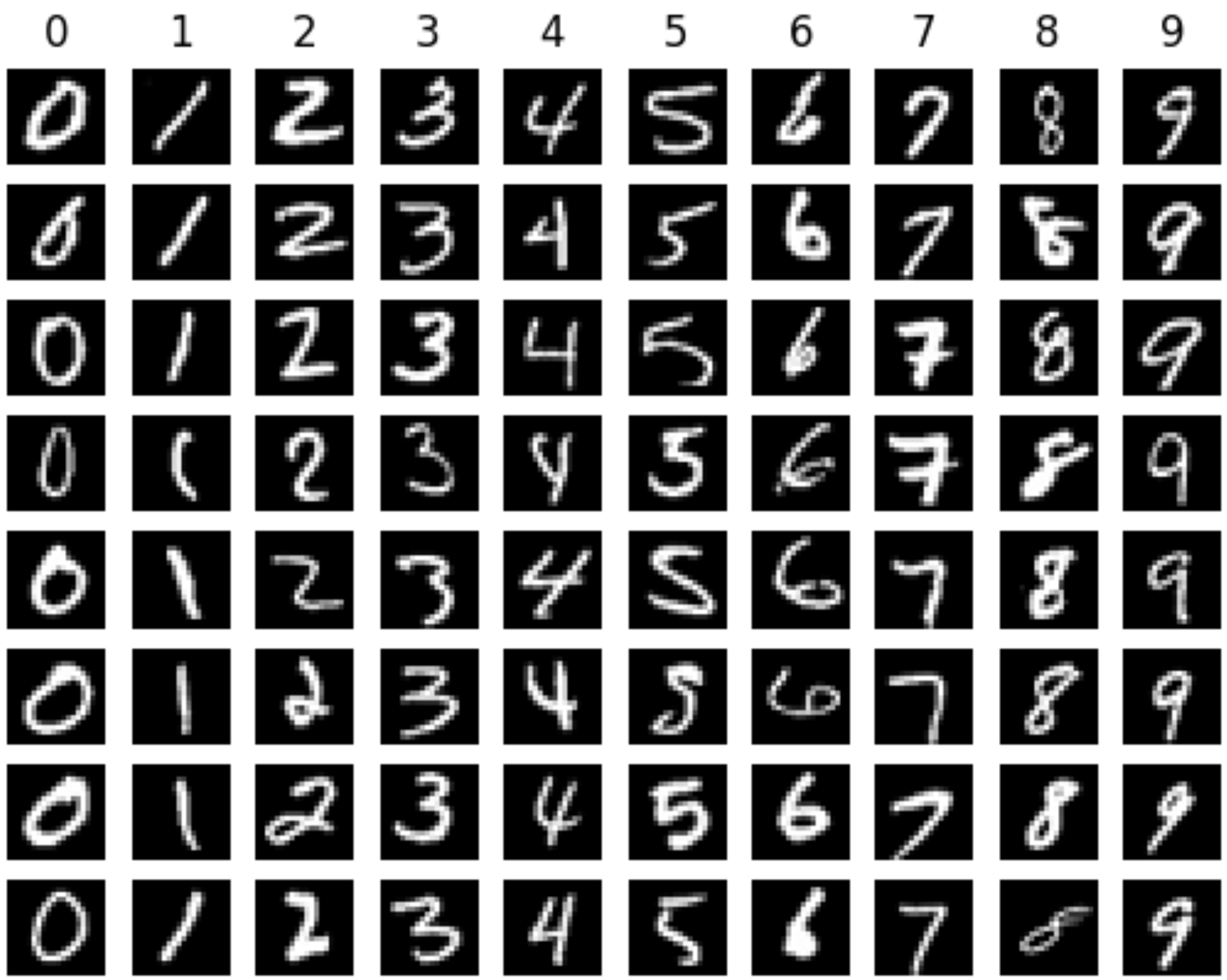


Which "leg-like" data point patterns were caused by a person's leg, which by furniture?

Or for MNIST-data:



Or for MNIST-data:



Which combinations of pixel values are most often seen for each of the numbers?
Which number is it that best explains the pixel values of a specific sample?

Some notations*

Some notations*

We express propositions as random variables taking on certain values directly

Some notations*

We express propositions as random variables taking on certain values directly

We look then for example at

$p(X = x_i)$, $i = 1, \ldots n$, for all n values x_i of the Variable X

E.g.: $p(Class = c_1) = p(Class = c_2) = \ldots = p(Class = c_n) = 1/10$

with c_i = "the image is showing the number 'i' "

Some notations*

We express propositions as random variables taking on certain values directly

We look then for example at

$p(X = x_i), i = 1,\ldots n$, for all n values x_i of the Variable X

E.g.: $p(Class = c_1) = p(\ Class = c_2) = \ldots = p(\ Class = c_n) = 1/10$

with c_i = "the image is showing the number 'i' "

For the *distribution* over the possible values of X we get then:

$p(\ X) = < p(X = x_1), p(\ X = x_2), \ldots, p(\ X = x_n) >$

i.e., when joining distributions, we iterate over a subset of the values for X and Y in the computation:

$p(\ X,Y) = p(\ X \mid Y)\ p(Y)$ describes a set of equations, expressing the joint probability distribution of X and Y as conditional probability distribution of X in dependency of the possible (or specifically given) values of Y

Some notations*

We express propositions as random variables taking on certain values directly

We look then for example at

$p(X = x_i), i = 1, \ldots n$, for all n values x_i of the Variable X

E.g.: $p(Class = c_1) = p(Class = c_2) = \ldots = p(Class = c_n) = 1/10$

with c_i = "the image is showing the number 'i' "

For the *distribution* over the possible values of X we get then:

$p(X) = < p(X = x_1), p(X = x_2), \ldots, p(X = x_n) >$

i.e., when joining distributions, we iterate over a subset of the values for X and Y in the computation:

$p(X,Y) = p(X \mid Y)\, p(Y)$ describes a set of equations, expressing the joint probability distribution of X and Y as conditional probability distribution of X in dependency of the possible (or specifically given) values of Y

Some notations*

We express propositions as random variables taking on certain values directly

We look then for example at

$p(X = x_i)$, $i = 1, \ldots n$, for all n values x_i of the Variable X

E.g.: $p(Class = c_1) = p(Class = c_2) = \ldots = p(Class = c_n) = 1/10$

with c_i = "the image is showing the number 'i' "

For the *distribution* over the possible values of X we get then:

$p(X) = < p(X = x_1), p(X = x_2), \ldots, p(X = x_n) >$

i.e., when joining distributions, we iterate over a subset of the values for X and Y in the computation:

$p(X,Y) = p(X \mid Y)\, p(Y)$ describes a set of equations, expressing the joint probability distribution of X and Y as conditional probability distribution of X in dependency of the possible (or specifically given) values of Y

Some notations*

We express propositions as random variables taking on certain values directly

We look then for example at

$p(X = x_i), i = 1,\ldots n$, for all n values x_i of the Variable X

E.g.: $p(Class = c_1) = p(Class = c_2) = \ldots = p(Class = c_n) = 1/10$

with c_i = "the image is showing the number 'i' "

For the *distribution* over the possible values of X we get then:

$p(X) = < p(X = x_1), p(X = x_2), \ldots, p(X = x_n) >$

i.e., when joining distributions, we iterate over a subset of the values for X and Y in the computation:

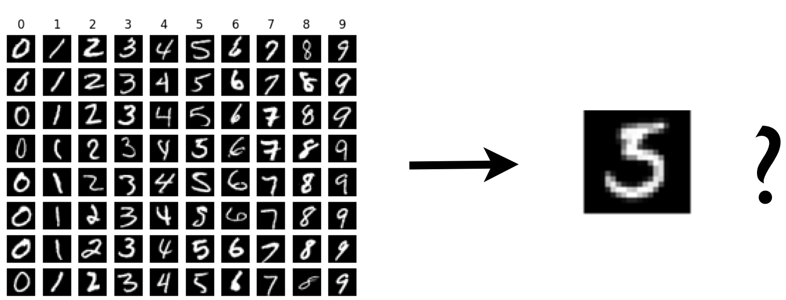
$p(X,Y) = p(X | Y) \, p(Y)$ describes a set of equations, expressing the joint probability distribution of X and Y as conditional probability distribution of X in dependency of the possible (or specifically given) values of Y

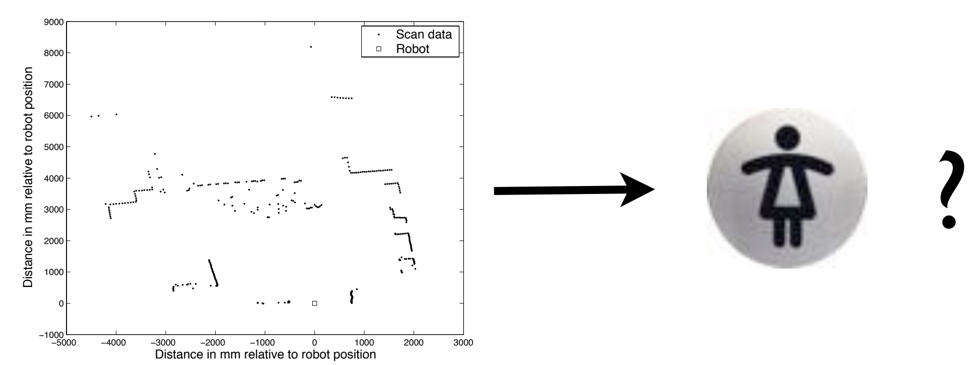
*) *There might be inconsistencies in the slides, anything that does not fit to this notation should be reported as faulty!*

Bayesian learning / classification

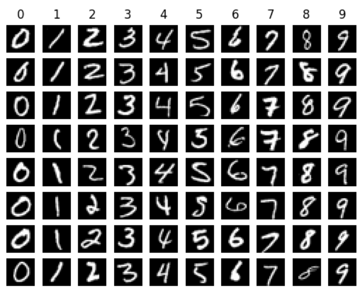
We want to classify / categorise / label new observations based on experience

More general: We want to *predict* and *explain* based on (limited) experience, to find categories / labels for observations or even the model for “how things work” (transition models, sensor models) *given a series of (explained) observations*.

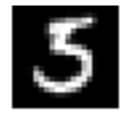




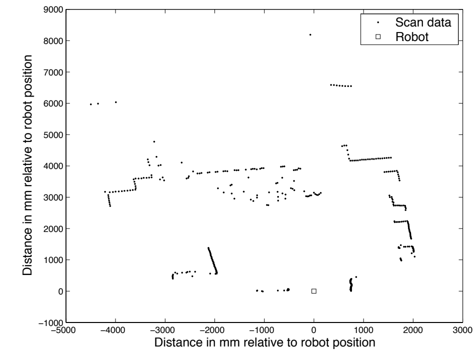
Bayesian classification



→



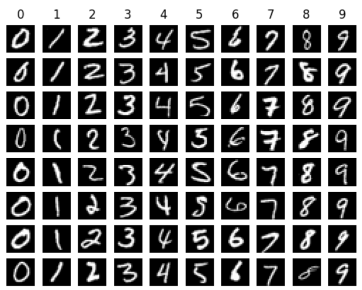
?

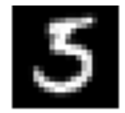


→

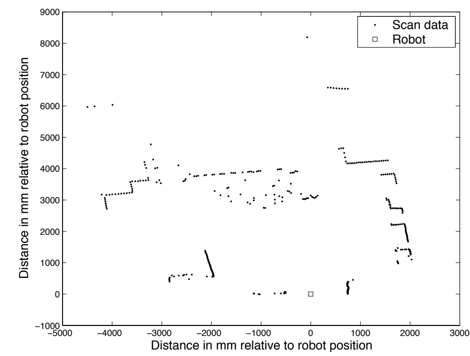
?

Bayesian classification



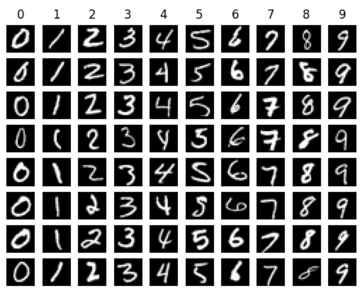


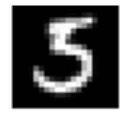
?



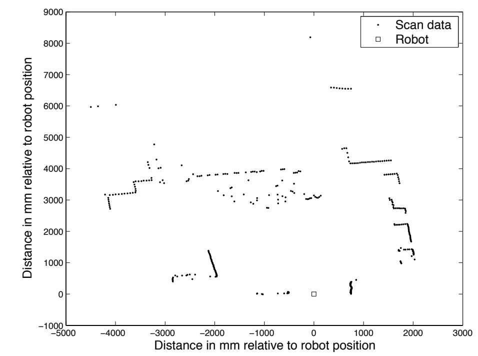
?

Bayesian classification



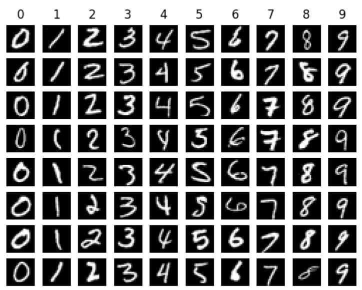


?

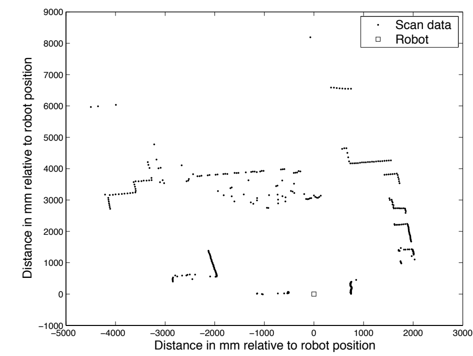


?

Bayesian classification

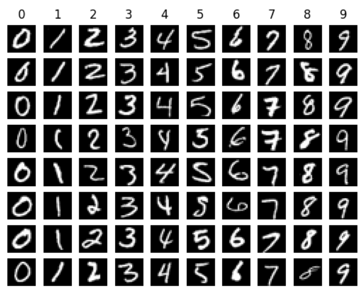


→  ?

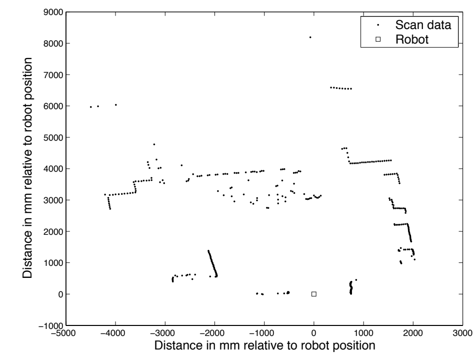


→  ?

Bayesian classification



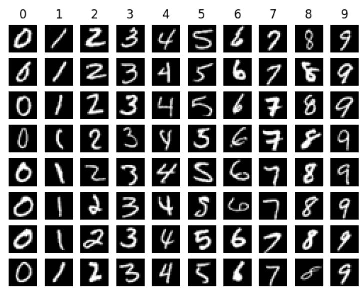
?



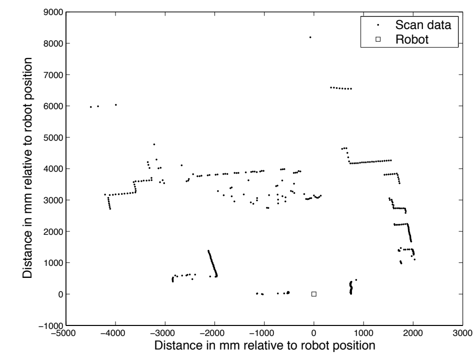
?

As a starting point we assume a known distribution for the data and want to do classification only:

Bayesian classification



?

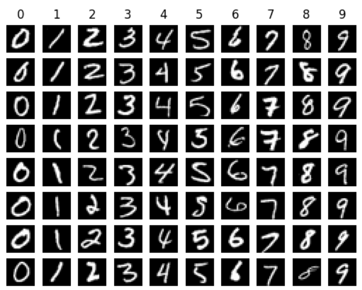


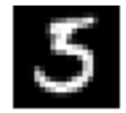
?

As a starting point we assume a known distribution for the data and want to do classification only:

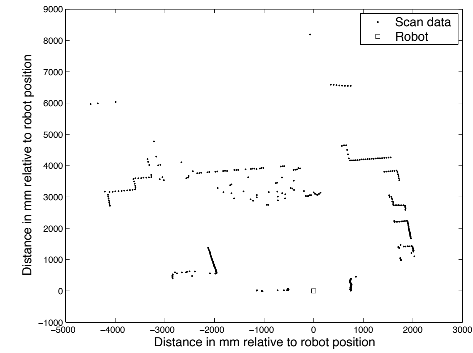
Ten classes $k = 0,\ldots,9$, i.e., an observed image shows a number 0,…,9 and belongs thus to the corresponding class k.
Each image X has 64 pixels x_i, $i = 0,\ldots,63$, that can take 17 values $v_j = 0,\ldots,16$, these we can call features or attributes.

Bayesian classification





?



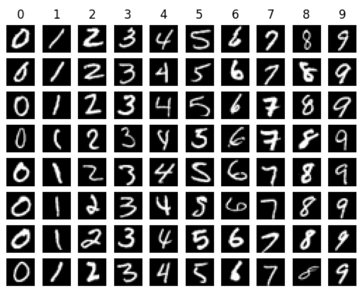
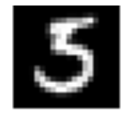
?

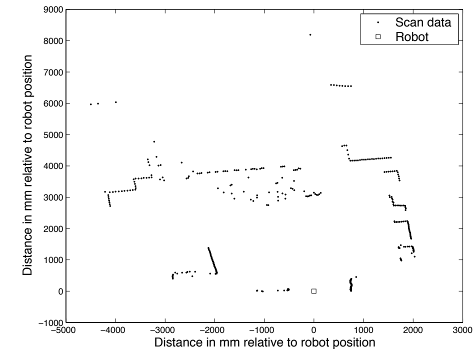
As a starting point we assume a known distribution for the data and want to do classification only:

Ten classes $k = 0,...,9$, i.e., an observed image shows a number 0,…,9 and belongs thus to the corresponding class *k*.
Each image *X* has 64 pixels x_i, $i = 0,...,63$, that can take 17 values $v_j = 0,...,16$, these we can call features or attributes.

or for the pattern problem:
Two classes of patterns, expressed through RV *Person* = {*person*, ¬*person*}, where each pattern has two features *LegSize* = {*legSize*, ¬*legSize*) and *Curved* = {*curved*, ¬*curved*}

Bayesian classification

  ?



 ?

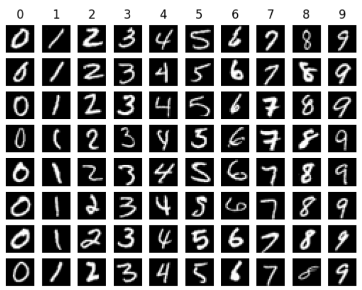
As a starting point we assume a known distribution for the data and want to do classification only:

Ten classes $k = 0,\ldots,9$, i.e., an observed image shows a number 0,…,9 and belongs thus to the corresponding class *k*.
Each image X has 64 pixels x_i, $i = 0,\ldots,63$, that can take 17 values $v_j = 0,\ldots,16$, these we can call features or attributes.

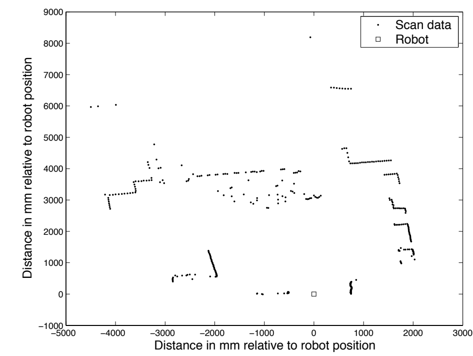
or for the pattern problem:
Two classes of patterns, expressed through RV *Person* = {*person*, ¬*person*}, where each pattern has two features *LegSize* = {*legSize*, ¬*legSize*) and *Curved* = {*curved*, ¬*curved*}

We produce some statistics over a given set of images, that tell us $p(x_i = v_j \mid k)$, or for the patterns $p(LegSize, Curved \mid Person)$

Bayesian classification



?



?

As a starting point we assume a known distribution for the data and want to do classification only:

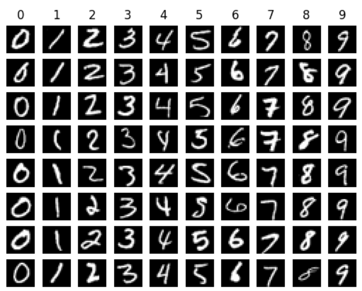
Ten classes $k = 0,\ldots,9$, i.e., an observed image shows a number 0,…,9 and belongs thus to the corresponding class *k*.
Each image X has 64 pixels $x_i, i = 0,\ldots,63$, that can take 17 values $v_j = 0,\ldots,16$, these we can call features or attributes.

or for the pattern problem:
Two classes of patterns, expressed through RV *Person* = {*person*, ¬*person*}, where each pattern has two features *LegSize* = {*legSize*, ¬*legSize*) and *Curved* = {*curved*, ¬*curved*}

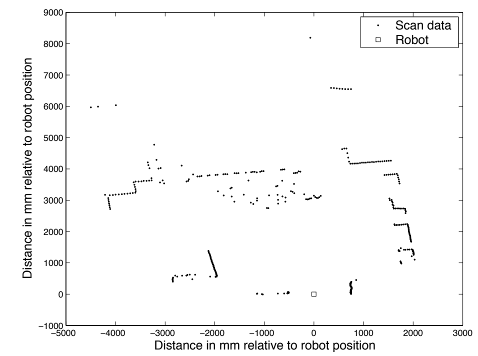
We produce some statistics over a given set of images, that tell us $p(x_i = v_j \,|\, k)$, or for the patterns $p(LegSize, Curved \,|\, Person)$

Now we make a new observation X_{new} of a combination of attributes (we see an image or pattern), and want to find the best hypothesis *h**, in this case directly corresponding to the class *k** or the value for *Person*.

Maximum Likelihood hypothesis

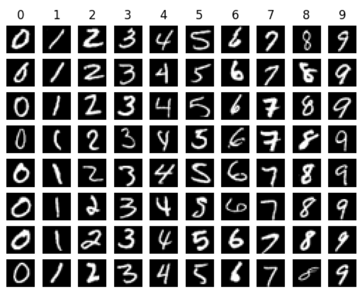


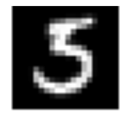
?



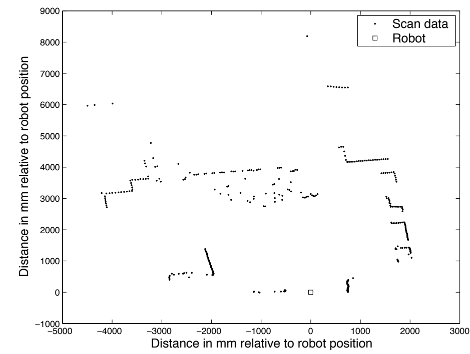
?

Maximum Likelihood hypothesis





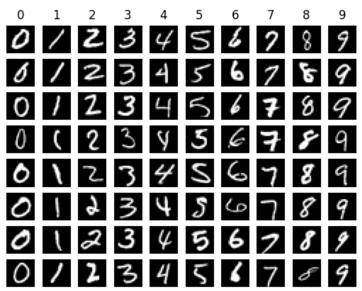
?



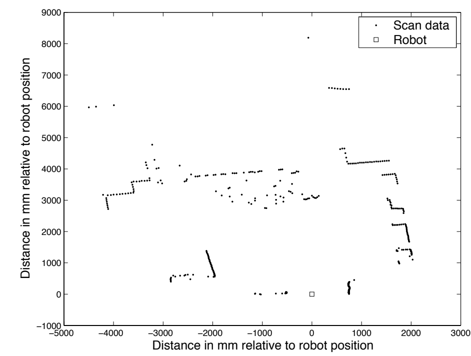
?

Use MLE, i.e., find the hypothesis / class that is best explaining the observation:

Maximum Likelihood hypothesis



?

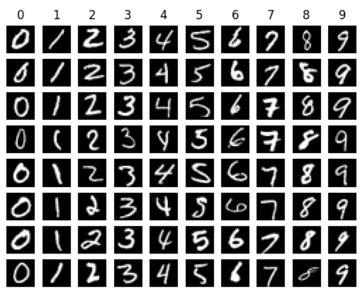


?

Use MLE, i.e., find the hypothesis / class that is best explaining the observation:

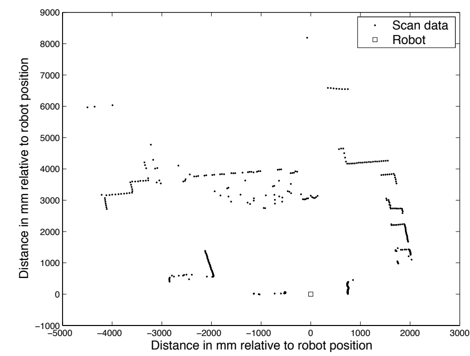
$$k^* = h_{ML} = \arg\max_k p(X_{new} \mid k)$$

Maximum Likelihood hypothesis



→

?



→

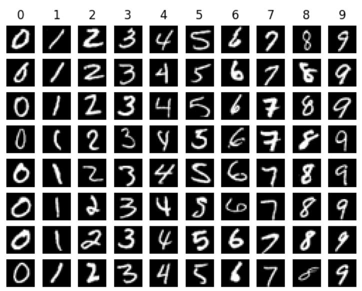
?

Use MLE, i.e., find the hypothesis / class that is best explaining the observation:

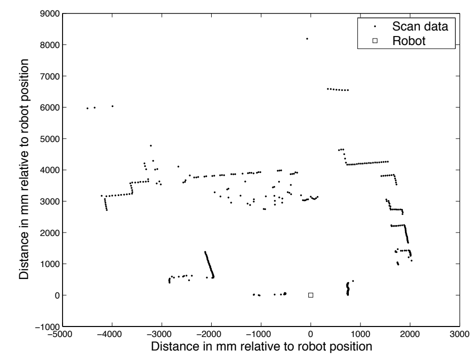
$$k^* = h_{ML} = \arg\max_k p(X_{new} \mid k)$$

Then apply this hypothesis to classify the image X_{new}: $Class(X_{new}) = k^*$

Maximum Likelihood hypothesis



→  ?



→  ?

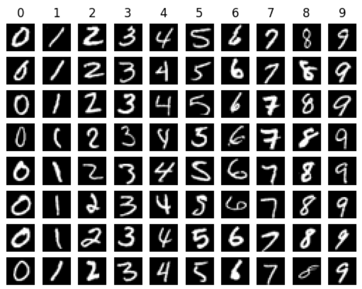
Use MLE, i.e., find the hypothesis / class that is best explaining the observation:

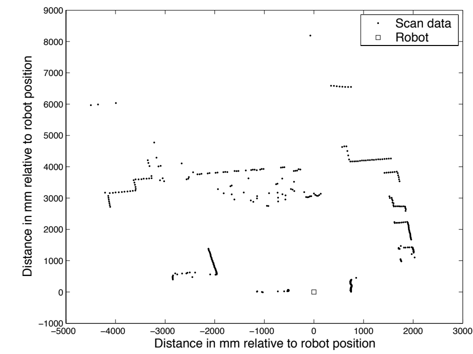
$$k^* = h_{ML} = \arg\max_k p(X_{new} \mid k)$$

Then apply this hypothesis to classify the image X_{new}: *Class(* X_{new}*)* = k^*

What if one type of image is much more prevalent in the "world" (data set)? Would that be reflected?

Maximum a Posteriori hypothesis


 ?



 ?

Use MAP, i.e., find the hypothesis / class that is best explained by the observation:

$$k^* = h_{MAP} = \underset{k}{\operatorname{argmax}}\, p(\, k \mid X_{new}) = \underset{k}{\operatorname{argmax}} \left[p(k)\, p(\, X_{new} \mid k) \right]$$

Then apply this hypothesis to classify the image X_{new}: *Class(* X_{new}*)* = k^*

Can we do even better? This seems still bold, in particular if there is little data. What if our assumption of the underlying distribution is not correct?

Will get back … first, there is another problem: how do we compute $p(\, X_{new} \mid k)$?

Remember: X_{new} is a combination of attribute values x_i, i.e., we want

$$p(\, X_{new} \mid k) = p(x_0 = v_0, x_1 = v_1, \ldots x_n = v_n \mid k)$$

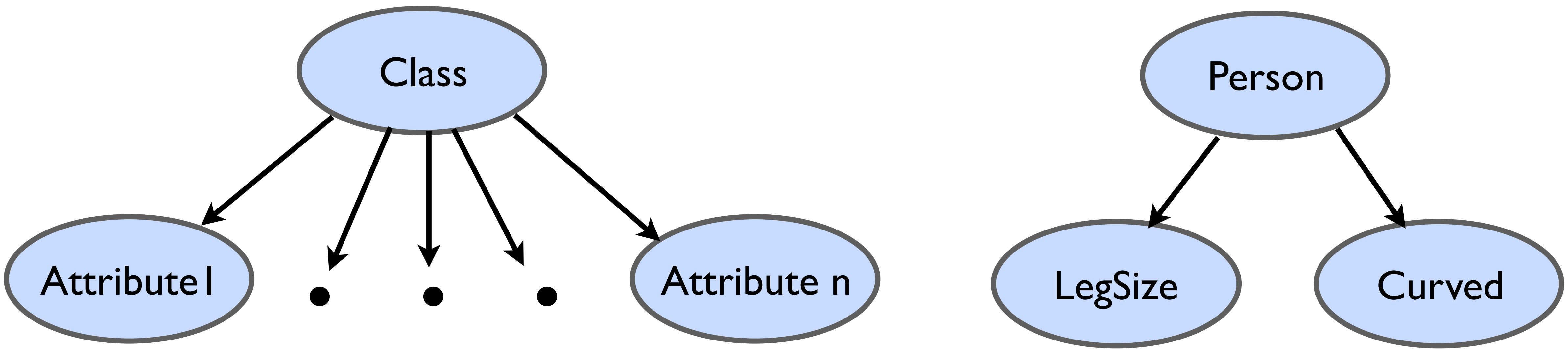
which becomes very complex with only few features, if we cannot assume independence.

Conditional independence in the MAP-approach

$$p(\,Class \mid Attribute1 = a1 \wedge Attribute2 = a2)$$
$$= \alpha \;\; p(\,a1 \wedge a2 \mid Class)\, p(\,Class)$$
$$= \alpha \;\; p(\,a1 \mid Class) p(\,a2 \mid Class)\, p(\,Class)$$

This gives a *naive Bayes* model (think Class = Cause, Attribute = Effect):

$$p(\,Class, Attribute_1, \ldots, Attribute_n) = \;\; p(\,Class) \prod_i p(\,Attribute_i \mid Class)$$



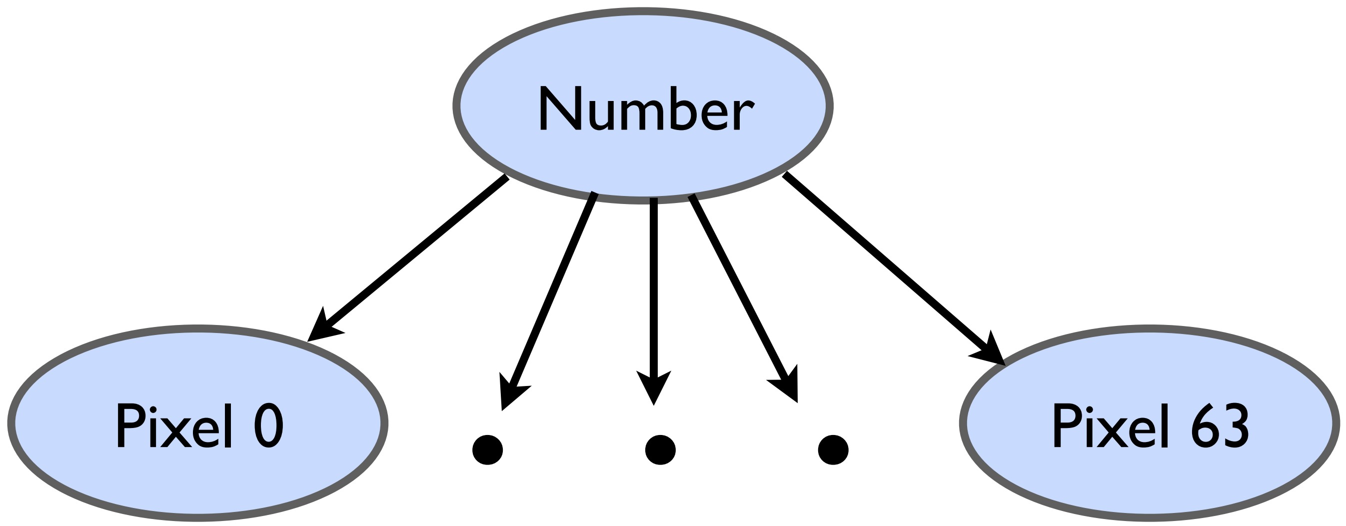
The total number of parameters is *linear* in *n*, and

$$p(Class = c*) = \underset{c}{\operatorname{argmax}}[p(Class = c, Attribute_0, \ldots, Attribute_{n-1})]$$

$$= \underset{c}{\operatorname{argmax}}[p(Class = c) \prod_i p(Attribute_i = av_j | Class = c)]$$

where av_j is the value that is observed for $Attribute_i$

A (super naïve) NBC for the digits data



$CPT_{0jk} = p(Pixel_0 = v_j | Number = k)$

$CPT_{63jk} = p(Pixel_{63} = v_j | Number = k)$

For a given (unknown) image X_{new} with $pixel_{new_i}$ representing the value of $Pixel_i$ in X_{new} we want to classify we get:

$$Number(X_{new}) = \arg\max_k [p(Number = k, pixel_{new_0}, \ldots, pixel_{new_{n-1}})]$$

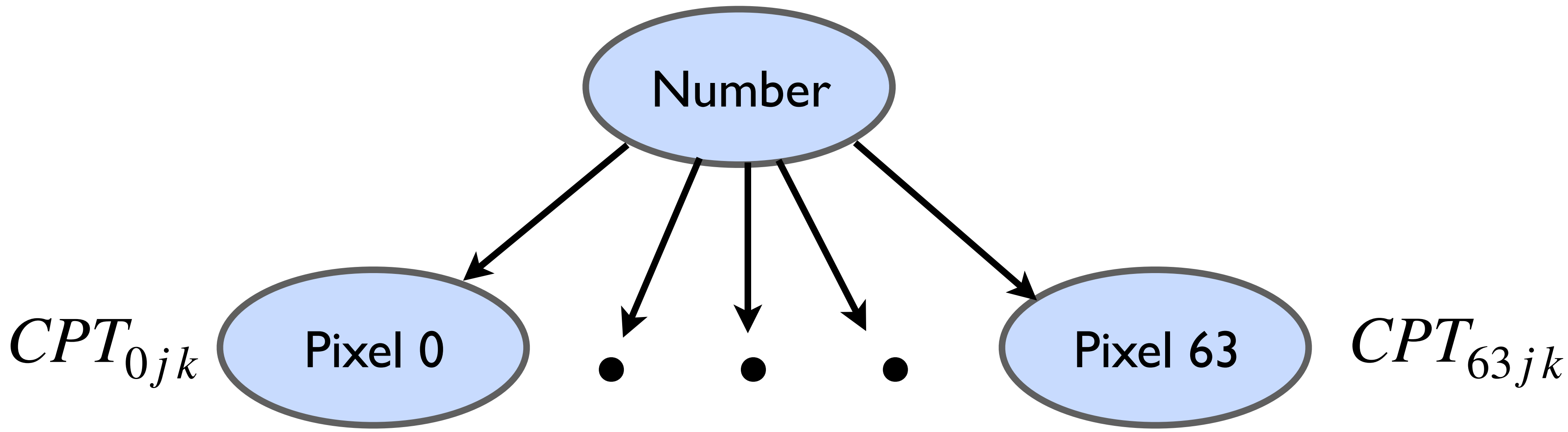
$$= \arg\max_k [p(Number = k) \prod_i p(x_{new_i} | Number = k)]$$

note: all possible attribute values are here the same for each attribute, hence v_j, not av_j

Today's agenda

- Recap conditional / posterior probabilities, Bayes' rule, independence / conditional independence
- The Naive Bayesian Classifier and the Gaussian Naive Bayesian Classifier
- Learning a Bayesian Classifier
- The unsupervised version: EM (for GMMs or k-Means)

Learning = “assigning” (?)



Simply set $CPT_{ijk} = p(pixel_i = v_j \,|\, Number = k) = \dfrac{|X_{i\,v_j k}|}{|X_k|}$

with

$X_{iv_jk} = \{examples\ X\ belonging\ to\ class/number\ k,\ where\ pixel\ i\ in\ X\ has\ value\ v_j\}$

What if…?

What if…?

$$X_{iv_jk} = \{examples\ X\ belonging\ to\ class/number\ k,\ where\ pixel\ i\ in\ X\ has\ value\ v_j\} = 0?$$

What if…?

$X_{iv_jk} = \{examples\ X\ belonging\ to\ class/number\ k, where\ pixel\ i\ in\ X\ has\ value\ v_j\} = 0?$

What if…?

$$X_{iv_jk} = \{examples\ X\ belonging\ to\ class/number\ k, where\ pixel\ i\ in\ X\ has\ value\ v_j\} = 0?$$

Then $CPT_{ijk} = p(pixel_i = v_j | Number = k) = \dfrac{|X_{i\,v_jk}|}{|X_k|} = 0$

What if…?

$$X_{iv_jk} = \{examples\ X\ belonging\ to\ class/number\ k, where\ pixel\ i\ in\ X\ has\ value\ v_j\} = 0?$$

Then $CPT_{ijk} = p(pixel_i = v_j | Number = k) = \dfrac{|X_{i\,v_jk}|}{|X_k|} = 0$

What if…?

$X_{iv_jk} = \{examples\ X\ belonging\ to\ class/number\ k,\ where\ pixel\ i\ in\ X\ has\ value\ v_j\} = 0?$

Then $CPT_{ijk} = p(pixel_i = v_j | Number = k) = \dfrac{|X_{i\,v_jk}|}{|X_k|} = 0$

This is "super naïve", but why is it actually bad?

“Padding” with m-estimate

“Padding” with m-estimate

Instead of $CPT_{ijk} = \dfrac{|X_{i\,v_jk}|}{|X_k|}$ use $CPT_{ijk} = \dfrac{|X_{i\,v_jk}| + mp}{|X_k| + m}$

“Padding” with m-estimate

Instead of $CPT_{ijk} = \dfrac{|X_{i\,v_j k}|}{|X_k|}$ use $CPT_{ijk} = \dfrac{|X_{i\,v_j k}| + mp}{|X_k| + m}$

“Padding” with m-estimate

Instead of $CPT_{ijk} = \frac{|X_{i\,v_j k}|}{|X_k|}$ use $CPT_{ijk} = \frac{|X_{i\,v_j k}| + mp}{|X_k| + m}$

where p is an estimate for the probability to observe attribute value v_j in the data (uniform is fine if you have no idea) and m is a confidence measure for this estimate grounded in the number of samples we expect this to happen for, $m = 1$ is fine for a start.

“Padding” with m-estimate

Instead of $CPT_{ijk} = \frac{|X_{i\,v_j k}|}{|X_k|}$ use $CPT_{ijk} = \frac{|X_{i\,v_j k}| + mp}{|X_k| + m}$

where p is an estimate for the probability to observe attribute value v_j in the data (uniform is fine if you have no idea) and m is a confidence measure for this estimate grounded in the number of samples we expect this to happen for, $m = 1$ is fine for a start.

Think of this as “adding a few invented samples”.

“Padding” with m-estimate

Instead of $CPT_{ijk} = \dfrac{|X_{i\,v_j k}|}{|X_k|}$ use $CPT_{ijk} = \dfrac{|X_{i\,v_j k}| + mp}{|X_k| + m}$

where p is an estimate for the probability to observe attribute value v_j in the data (uniform is fine if you have no idea) and m is a confidence measure for this estimate grounded in the number of samples we expect this to happen for, $m = 1$ is fine for a start.

Think of this as “adding a few invented samples”.

Technically, the m-estimate in itself can be learned from / adapted to the data in the process.

Does that really work well?

Does that really work well?

- The digits data set has images with 8x8 = 64 pixels with discrete values in the range [0, …, 16]

Does that really work well?

- The digits data set has images with 8x8 = 64 pixels with discrete values in the range [0, …, 16]
- A more realistic "MNIST_Light data" set has 20x20 = 400 pixels with (discrete) values in the range [0.0, …, 255.0]

Does that really work well?

- The digits data set has images with 8x8 = 64 pixels with discrete values in the range [0, …, 16]
- A more realistic "MNIST_Light data" set has 20x20 = 400 pixels with (discrete) values in the range [0.0, …, 255.0]
- Is there a more general way to express how much an example belongs to a class?

Does that really work well?

- The digits data set has images with 8x8 = 64 pixels with discrete values in the range [0, …, 16]
- A more realistic "MNIST_Light data" set has 20x20 = 400 pixels with (discrete) values in the range [0.0, …, 255.0]
- Is there a more general way to express how much an example belongs to a class?

Does that really work well?

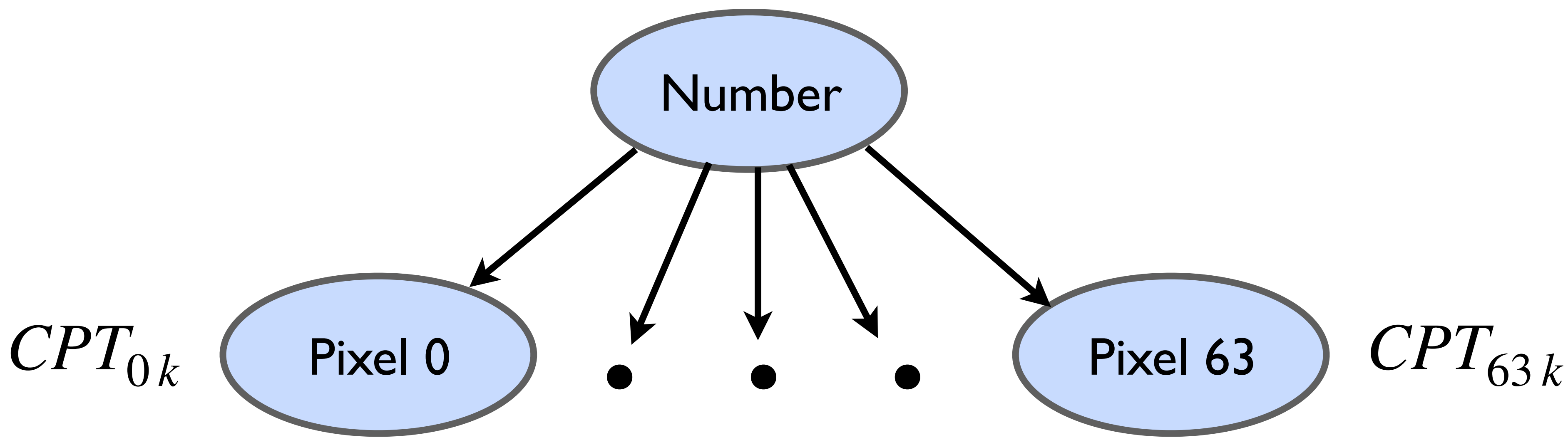
- The digits data set has images with 8x8 = 64 pixels with discrete values in the range [0, …, 16]
- A more realistic "MNIST_Light data" set has 20x20 = 400 pixels with (discrete) values in the range [0.0, …, 255.0]
- Is there a more general way to express how much an example belongs to a class?

- "Blur" the probabilities by using a suitable distribution (often, not always, a Gaussian Normal Distribution can help)

Gaussian Mixture Model

- Assume that the *n* attributes in the example set form the axes of an *n*-dimensional feature space, i.e., each example is a "point" in that space.
- The examples belonging to a class will then somehow "gather" around some centre "point"
- The degree of "belonging" can be expressed as a continuous PDF - very often a Gaussian Normal distribution is suitable, which gives then a Gaussian Mixture Model (the multidimensional bell "curves" will most likely overlap, hence, there is a mixture of several distributions that explain a given data point - the sample to be classified).
- see Goodfellow (3) / Murphy (2) for other "standard" distributions

Gaussian Naive Bayesian Classifier



$$CPT_{ik} = (\mu_{ik}, \sigma_{ik}), \quad p(pixel_i = x \,|\, Number = k) = \frac{1}{\sqrt{2\pi\sigma_{ik}^2}} e^{-\frac{1}{2\sigma_{ik}^2}(x-\mu_{ik})^2}$$

with $\mu_{ik} = mean(x_i)$ and $\sigma_{ik}^2 = var(x_i)$ $\forall X \in \{examples\ where\ Number = k\}$

and x_i the value of *Pixel i* in a given image X

Classification is then handled for unseen sample X_{new}:

$$Number(X_{new}) = \underset{k}{\operatorname{argmax}} \left[p(Number = k, x_{new_0}, \ldots, x_{new_{n-1}})\right]$$

$$= \underset{k}{\operatorname{argmax}} \left[p(Number = k) \prod_i p(x_{new_i}) \,|\, Number = k)\right]$$

Today's agenda

- Recap conditional / posterior probabilities, Bayes' rule, independence / conditional independence
- The Naive Bayesian Classifier and the Gaussian Naive Bayesian Classifier
- Learning a Bayesian Classifier
- The unsupervised version: EM (for GMMs or k-Means)

EM-algorithm for GMMs

This is the EM-algorithm as given by Murphy, "Machine Learning - A Probabilistic Approach", p 353.
assume data set $\mathbf{X}$ with examples $\overrightarrow{x_i}, i = 1,\ldots,N$ and K classes you want to cluster $\mathbf{X}$ into.

EM-for-GMM($\mathbf{X}, \mathbf{K}$)

1. Initialize $\theta_k^0 = (\pi_k^0, \overrightarrow{\mu_k}^0, Cov_k^0)$, where
   π_k is the class prior for class *k* (e.g., assume uniform distribution here initially)
   $\overrightarrow{\mu_k}$ are the means for the attribute values per attribute *a* in class *k* (initially a random subset of $\mathbf{X}$)
   Cov_k is the covariance for the attribute values in class *k* (can be simplified to variance σ_{ak}^2 for each attribute *a* if a G-NBC is assumed as the model)
2. Iterate over E and M steps as follows:

   E-step:

   compute $r_{ik}^t = \dfrac{\pi_k^{t-1} p(\vec{x}_i \mid \theta_k^{t-1})}{\sum_{k'} \pi_{k'} p(\vec{x}_i \mid \theta_{k'}^{t-1})}$ where $p(\vec{x}_i \mid \theta_k^{t-1}) = \prod_a \dfrac{1}{\sqrt{2\pi\sigma_{ak}^2}} e^{-\frac{1}{2\sigma_{ak}^2}(x-\mu_{ak}^{t-1})^2}$, assuming that

   the covariance can be substituted with σ_{ak}^2 for attribute *a* and class *k*. Index *i* runs over the samples, index *a* runs over the attributes, and *k* over clusters ("classes")

   M-step:

   compute $r_k^t = \sum_i r_{ik}^t \quad and \quad \pi_k^t = \dfrac{r_k^t}{N}$, then update the means and variances:

   $\overrightarrow{\mu_k}^t = \dfrac{\sum_i r_{ik}^t \overrightarrow{x_i}}{r_k^t} \quad and \quad Cov_k^t = \dfrac{\sum_i r_{ik}^t \vec{x}_i \vec{x}_i^T}{r_k^t} - \overrightarrow{\mu_k^t}\overrightarrow{\mu_k^t}^T$ (from which the new σ_{ak}^2 can be extracted)

3. Stop, when the $\overrightarrow{\mu_k}$ and Cov_k are not changing significantly anymore.

EM-algorithm for k-Means

This is the EM-algorithm as given by Murphy, "Machine Learning - A Probabilistic Approach", p 356.
assume data set $\mathbf{X}$ with examples $\overrightarrow{x_i}, i = 1,\ldots,N$ and K classes you want to cluster $\mathbf{X}$ into.

k-Means($\mathbf{X}$, K)
1. Initialize $\overrightarrow{\mu_k}^0$, assume fixed class priors π_k
2. *Iterate* over E and M steps as follows:

E-step:

Assign each data point to its closest cluster centre: $z_i = \underset{k}{\operatorname{argmin}} \| \vec{x}_i - \overrightarrow{\mu}_k \|_2^2 = L_2(\vec{x}_i - \overrightarrow{\mu}_k)^2$

M-step:

Update each cluster centre by computing the means of all points assigned to it:

$$\overrightarrow{\mu}_k = \frac{1}{N_k} \sum_{i:z_i=k} \vec{x}_i$$

Until converged

Today's summary

- (Refreshed memory on conditional probabilities, Bayes' rule, independence, conditional independence)
- Introduced Naive Bayesian Classifiers
- Introduced GNBCs (very briefly)
- Rushed through the EM-algorithm…

- Reading:
  - Goodfellow, ch 3, Murphy, ch 2
  - Lecture slides lecture 11, 2018
  - Mitchell, chapter 6