Machine Learning

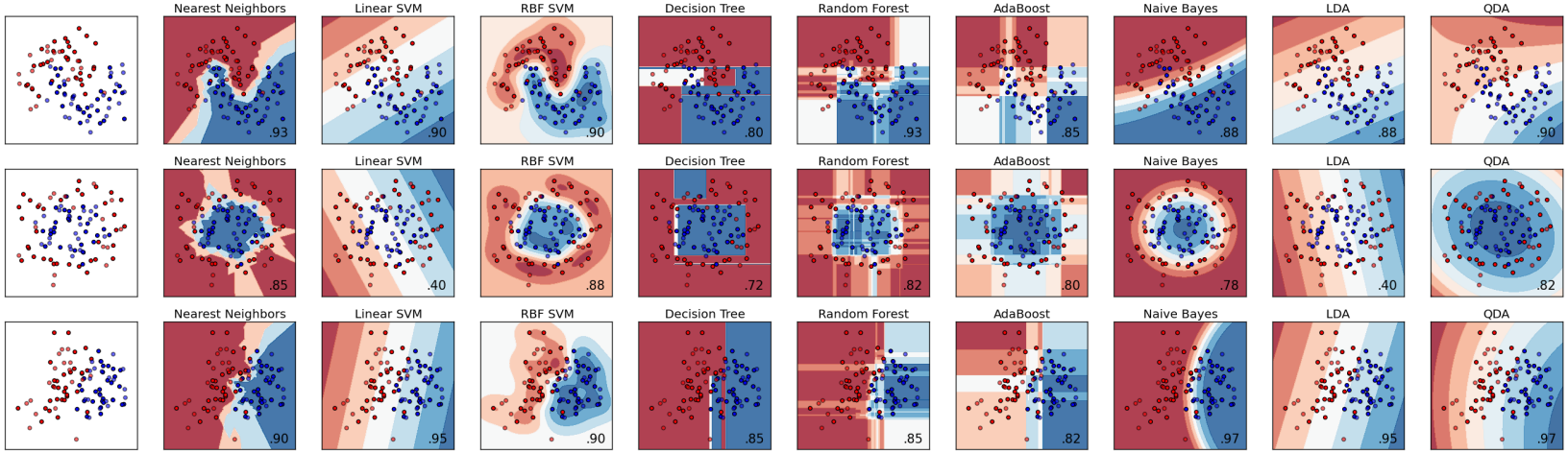
– winter term 2016/17 –

Chapter 11: Naive Bayes and Graphical Models

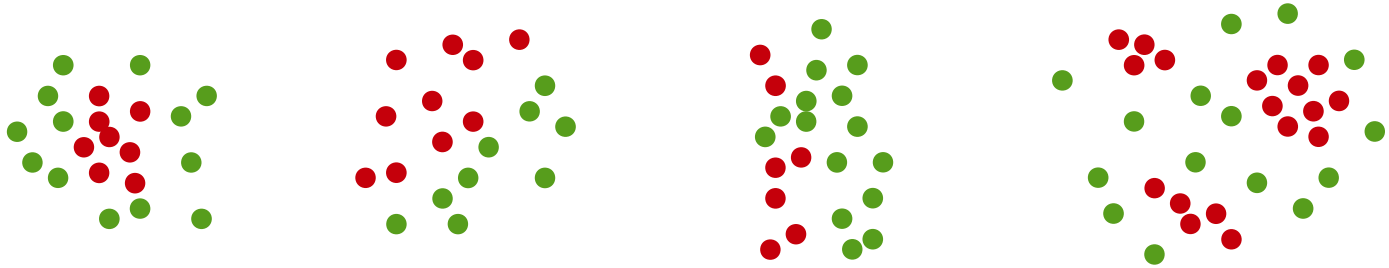
Prof. Adrian Ulges
Masters "Computer Science"
DCSM Department
RheinMain University of Applied Sciences

Outline

The No-free-Lunch Theorem image from [1]



- We have seen different classifiers now
- They all have their **benefits** and **drawbacks** *(SVMs are preferred for small training sets, decision trees when few features can lead to an accurate decision, ...)*
- **Key Question**: Is there a classifier that is **best** when averaging over **many/all learning problems**?



The No-free-Lunch Theorem

> *"If one is interested in* **off-training-set error***, then there are no a priori distinctions between learning algorithms. All algorithms are* **equivalent, on average***."*

(www.no-free-lunch.org)

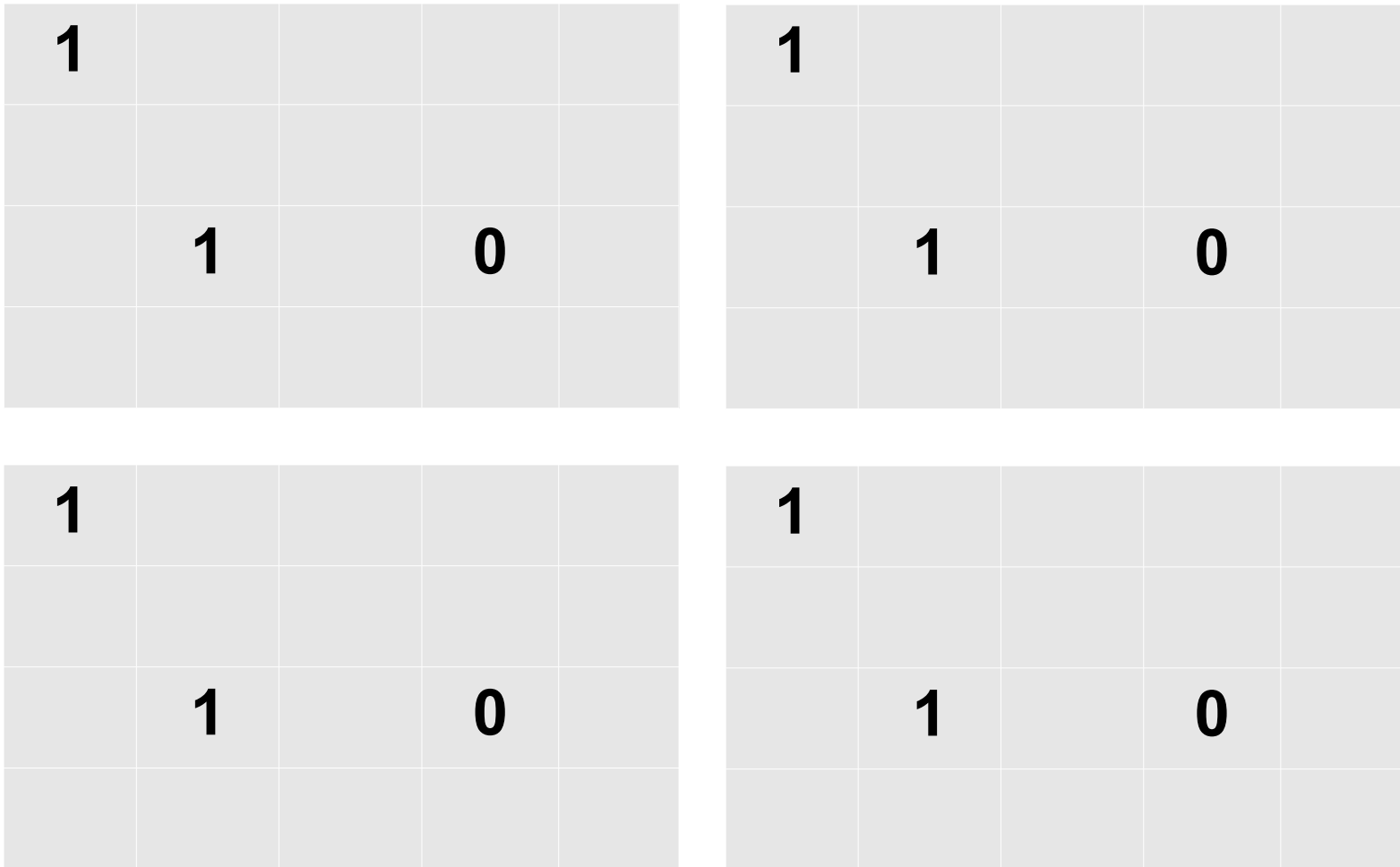
> *"For any two algorithms A and B, there are 'as many' targets (or priors over targets) for which A has lower expected OTS (off-training-set) error than B as vice versa."*

(Wolpert 1997)

Remarks

- **This means**: If a classifier performs well on a certain set of problems, then it necessarily performs worse on the set of other problems.
- A *target* is a learning problem.
- The theorem does not take into account that some targets are **more likely** than others.

No Free Lunch: Illustration



No Free Lunch

Outline

Machine Learning and Bayes' Rule

- Our **goal**: Compute the **posterior** $P(c|\mathbf{x})$.
- **Observation**: We can 'turn around' the posterior using Bayes' rule

"This is the "most important equation" in machine learning."

(S. Marsland)

Discriminative vs. Generative Models

In general, there are two general **kinds of classifiers**

1. **Discriminative** methods: use a direct model for $P(c|\mathbf{x})$
   (or, alternatively, the decision boundary)
2. **Generative** methods: compute $P(c)$ and $P(\mathbf{x}|c)$
   and plug them into Bayes' rule

Generative Methods

- $\mathbf{P(c)}$ is easy to compute: We estimate each **class's frequency**.
  - 'two of three e-mails are spam' $\rightarrow P(\text{spam}) = 0.1$
  - 'men and woman are equally frequent' $\rightarrow P(0) = P(1) = 0.5$
- $P(\mathbf{x}|c)$ is a bit more **tricky**
  - for discrete features: $P(\mathbf{x}|c)$ is a probability table
  - for continuous features: $P(\mathbf{x}|c)$ is a probability density

Generative Methods: Do-it-Yourself

We build a classifier for cars, with **classes** *cheap*, *medium-priced* and *costly*. Our **features** are *color* and *PSs*.

- Turn the values in the below tables into probabilities. Where are the priors, where the CCDs?
- Classify a red car with few PSs.

class 'cheap' (2000 cars)

PSs / color	red	blue	silver
lots	200	40	160
few	800	300	500

class 'medium' (1000 cars)

PSs / color	red	blue	silver
lots	130	70	200
few	470	60	70

class 'costly' (500 cars)

PSs / color	red	blue	silver
lots	150	110	100
few	60	20	60

Generative Methods: Do-it-Yourself

Generative Methods: Do-it-Yourself

Generative Methods: Naive Bayes

The Curse of Dimensionality

In case **x** is high-dimensional, the class-conditional densities become increasingly **difficult to learn**.

- Let **x** be a boolean vector with n entries, i.e. $\mathbf{x} = (x_1, ..., x_n)$
- $P(\mathbf{x}|c)$ is a probability table with 2^n entries
- **Example: Spam filtering**
  - 2 classes, spam and ham
  - n=2 boolean features: x_1 (is the sender of the e-mail known?) and x_2 (does the e-mail contain the term 'viagra'?)

class 1 (SPAM)			class 2 (HAM)		
$\mathbf{x}_1$(*sender*)	$\mathbf{X}_2$(*„Viagra“*)	P(x_1,x_2|spam)	$\mathbf{x}_1$(*sender*)	$\mathbf{X}_2$(*„Viagra“*)	P(x_1,x_2|ham)
0	0	0.78	0	0	0.18
0	1	0.20	0	1	0.02
1	0	0.01	1	0	0.78
1	1	0.02	1	1	0.02

20% of all SPAM mails come from unknown senders and contain the term 'viagra'.

Generative Methods: Naive Bayes

- ... when n becomes large, we cannot learn all 2^n entries reliably ($\rightarrow$ *curse of dimensionality*)
- In spam filtering: Let **x** is a vector with $10,000$ entries *(which terms appear in the e-mail, which do not?)*
- The probability tables holds $2^{10,000}$ entries!

n=1

„Viagra“	P(x|spam)
0	0.78
1	0.22

n=2

„Viagra“	„nigeria“	P(x|spam)
0	0	0.58
0	1	0.22
1	0	0.16
1	1	0.04

n=3

„Viagra“	„nigeria“	„mom“	P(x|spam)
0	0	0	0.35
0	0	1	0.13
0	1	0	0.17
0	1	1	0.03
1	0	0	0.19
1	0	1	0.03
1	1	0	0.09
1	1	1	0.01

... **n=10,000** ?

We need to learn ***each*** of these entries from a (limited) training set!

Naive Bayes: Derivation

Idea of Naive Bayes: We simplify $P(\mathbf{x}|c)$ by assuming that the entries in the vector are *independent* (thus *Naive* Bayes)

Naive Bayes: Derivation

Naive Bayes

Remarks

- The number of entries to be learned decreases from 2^n to $2n$.
- We can estimate these $2n$ entries, even from limited-size training sets.
- This principle works for **any** form of CCDs!

Naive Bayes: Do-it-Yourself

Exercise the above **car example** using Naive Bayes!

class 'cheap' (2000 cars)

PSs / color	red	blue	silver
lots	200	40	160
few	800	300	500

class 'medium' (1000 cars)

PSs / color	red	blue	silver
lots	130	70	200
few	470	60	70

class 'costly' (500 cars)

PSs / color	red	blue	silver
lots	150	110	100
few	60	20	60

Naive Bayes: Real-valued Features

$$P(\mathbf{x}|c) = P(x_1|c) \cdot P(x_2|c) \cdot ... P(x_d|c)$$

- So far, we have studied Naive Bayes for **discrete features**
- Here, the CCDs $P(x_i|c)$ are **probability tables**
- However, this principle works for **any** form of CCDs!

Naive Bayes for real-valued Features

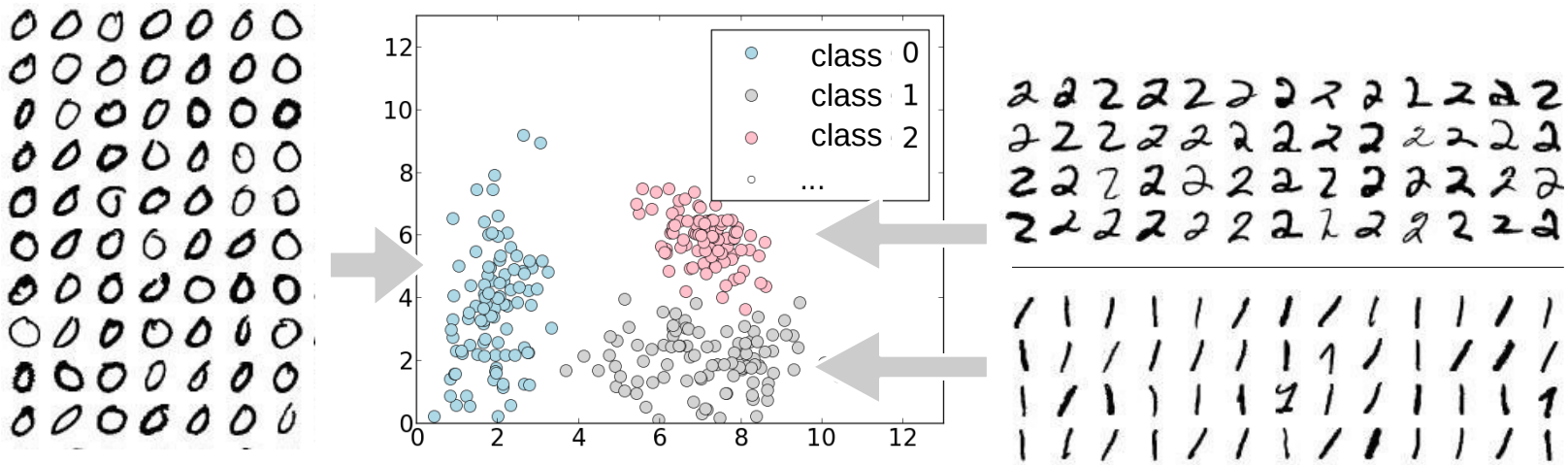
- For real-valued features, $P(x_i|c)$ becomes a **probability density function** $p(x_i|c)$
- **Examples**: (multivariate) normal distribution, uniform distribution, exponential distribution, ...

$$p(\mathbf{x}|c) = p(x_1|c) \cdot p(x_2|c) \cdot ... \cdot p(x_d|c)$$

- We will have a look at one example in the following.

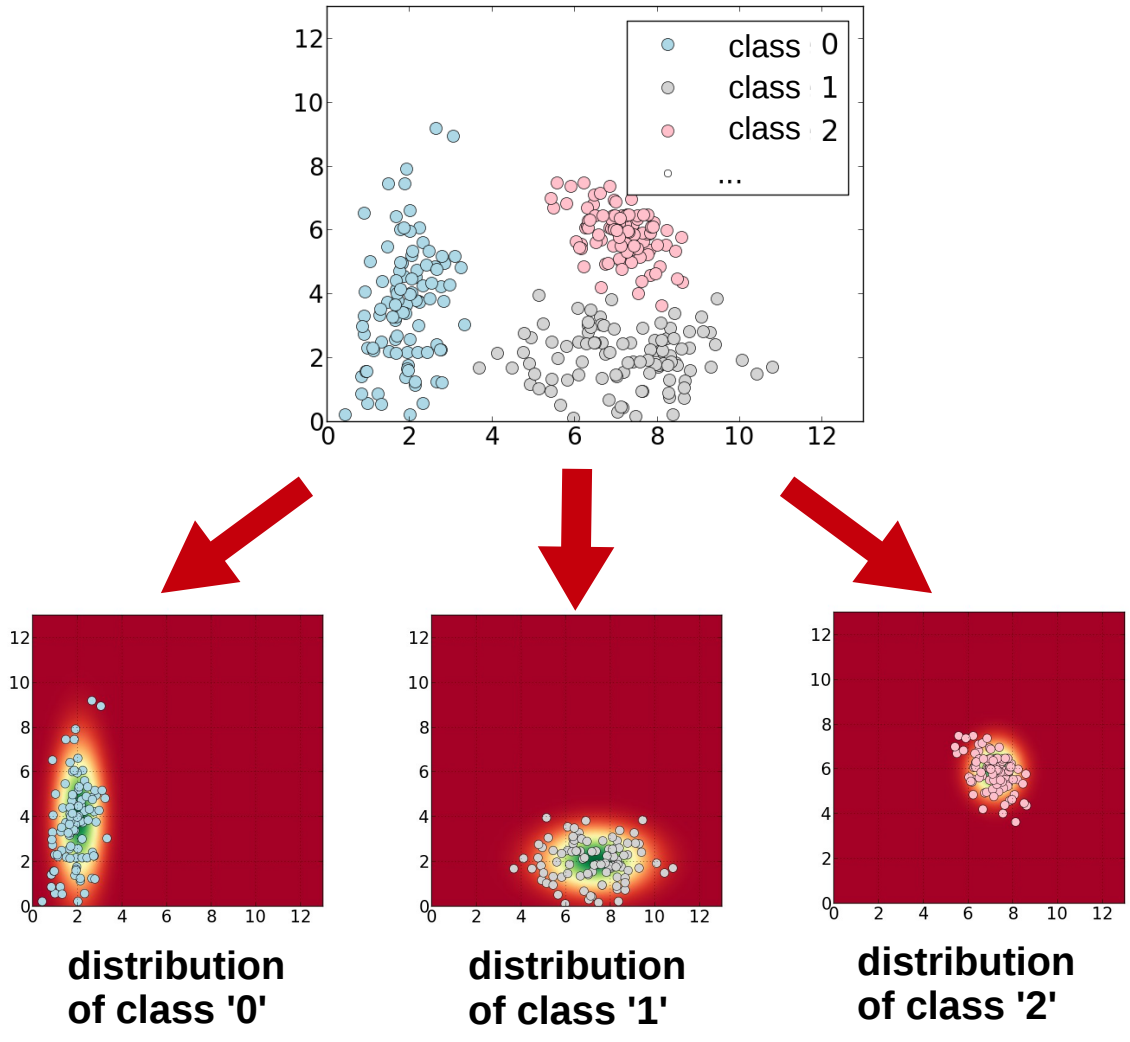
Naive Bayes Example: OCR

- OCR = "Optical Character Recognition"
  (*here: handwriting recognition*)
- Given the picture of a letter, decide which letter is visible
- Simple **feature vector**: scale the image to 20×20 pixels and store the intensity values into a vector $\mathbf{x} \in \mathbb{R}^{400}$
- What might be a suitable distribution for $p(\mathbf{x}|c)$?



OCR Training (Illustration)

Training = Estimating the classes' parameters $\mu_0, \mu_1, ..., \mu_9$ and $\Sigma_0, \Sigma_1, ..., \Sigma_9$



OCR: Naive Bayes

Training

- We apply Maximum Likelihood (ML) estimation for all classes, and obtain parameter estimates $\hat{\mu}_0, ..., \hat{\mu}_9$ and $\hat{\Sigma}_0, ...\hat{\Sigma}_9$.
- This way, we have learned the **distribution of all classes** (i.e., digits) in feature space.

Training with Naive Bayes?

- Remember that we want to use **naive Bayes**, i.e. we assume the single entries of the feature vectors are **independent**!
- What does this mean for our classifier? $\rightarrow$ the covariance matrix is a diagonal matrix!

$$\Sigma = \begin{pmatrix} \sigma_1^2 & 0 & ... & 0 \\ 0 & \sigma_2^2 & ... & 0 \\ ... & & & \\ 0 & 0 & ... & \sigma_d^2 \end{pmatrix}$$

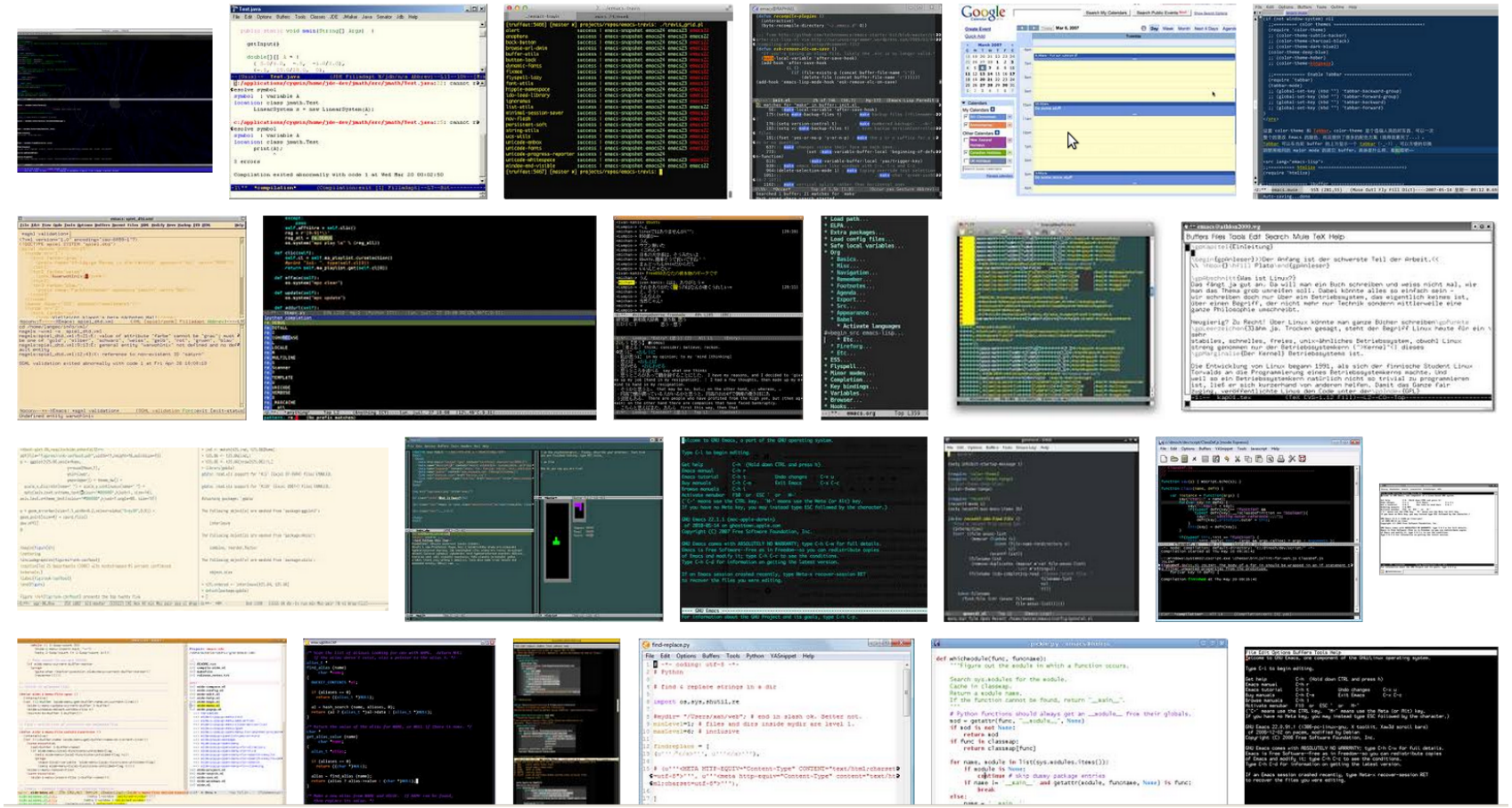
OCR: Applying our classifier

To recognize a new **digit x**, we apply our Naive Bayes classifier:

OCR: Applying our classifier

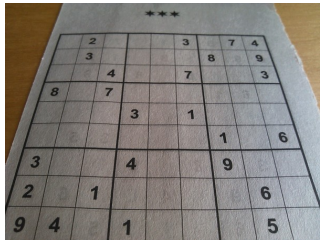
We choose the class (digit) c for which $P(c|\mathbf{x})$ is maximal:

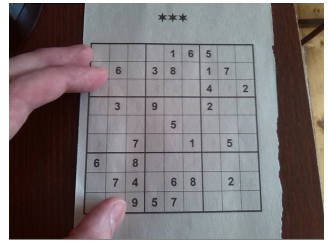
OCR: Code Sample



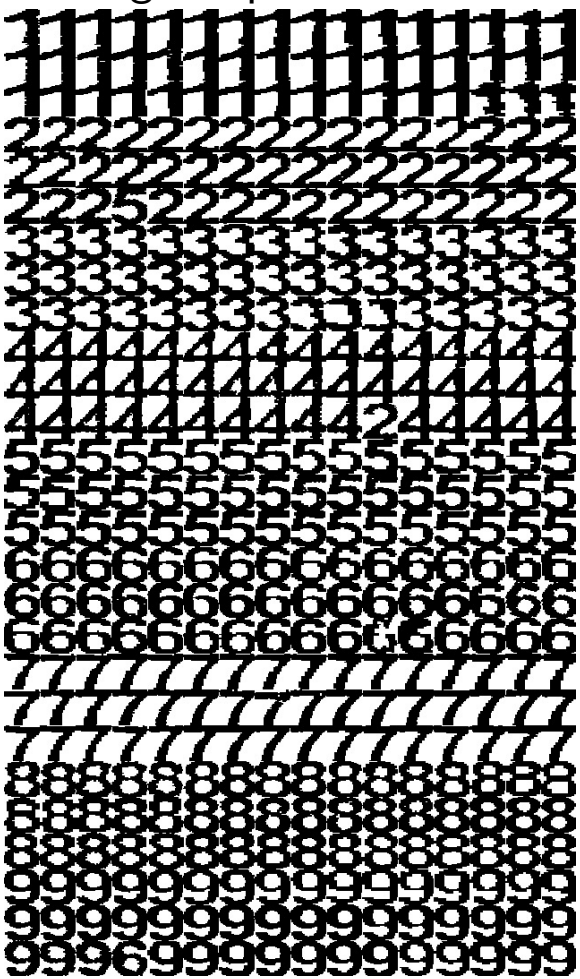
OCR: Sample Result

SUDOKU Camera[1]

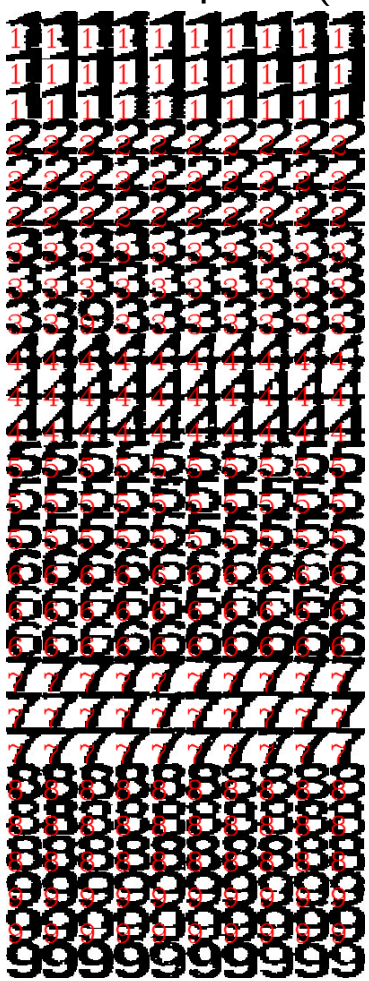




training samples



test samples (incl. results)

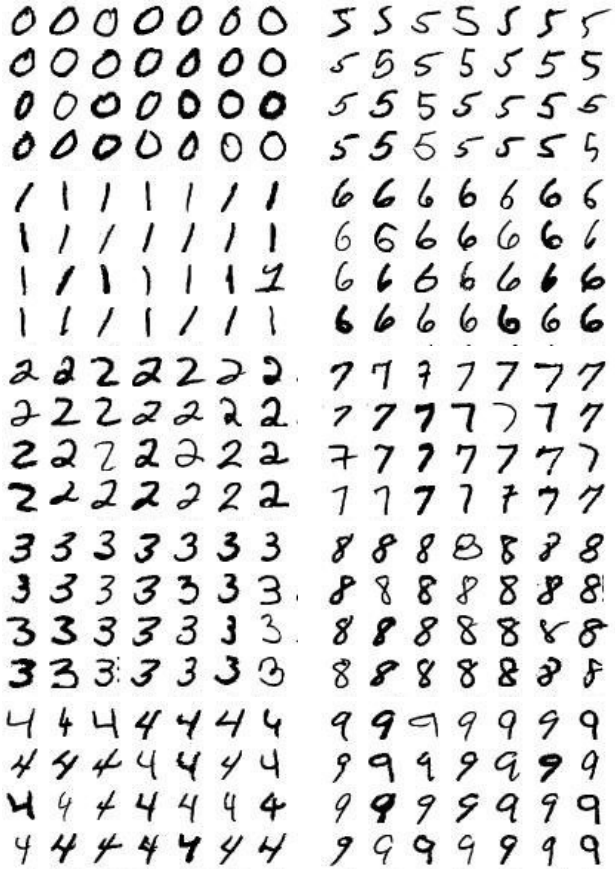


[1] *data supplied by Mattis Rehmke – kudos!*

OCR: Sample Result (cont'd)

handwriting recognition[2]

training set



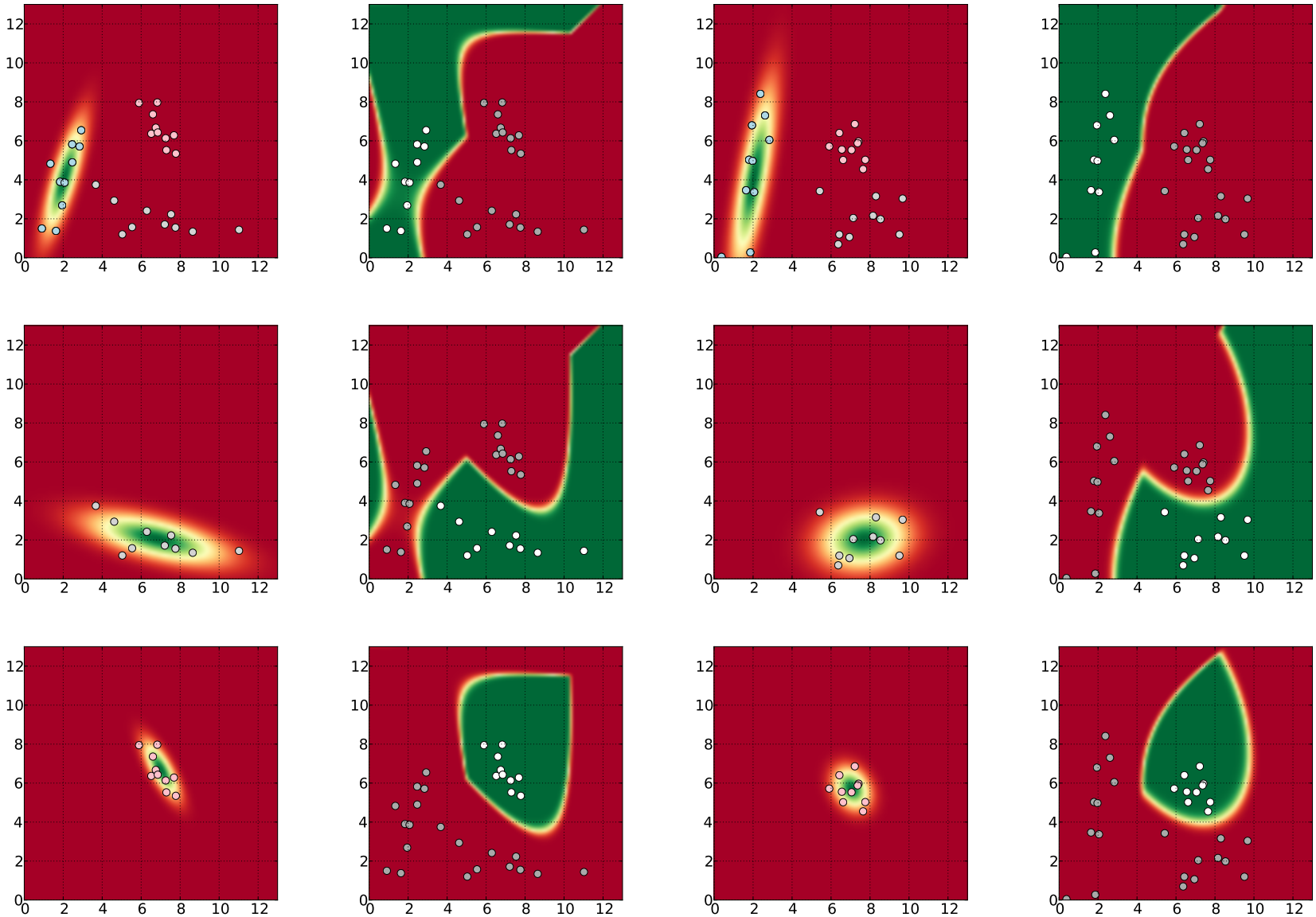
test set (incl. results)

[2] MIST Handwritten Digits Database: http://yann.lecun.com/exdb/mnist/

Overfitting in Naive Bayes

Run 1 *(10 samples per class)*

(CCDs left, posterior right)

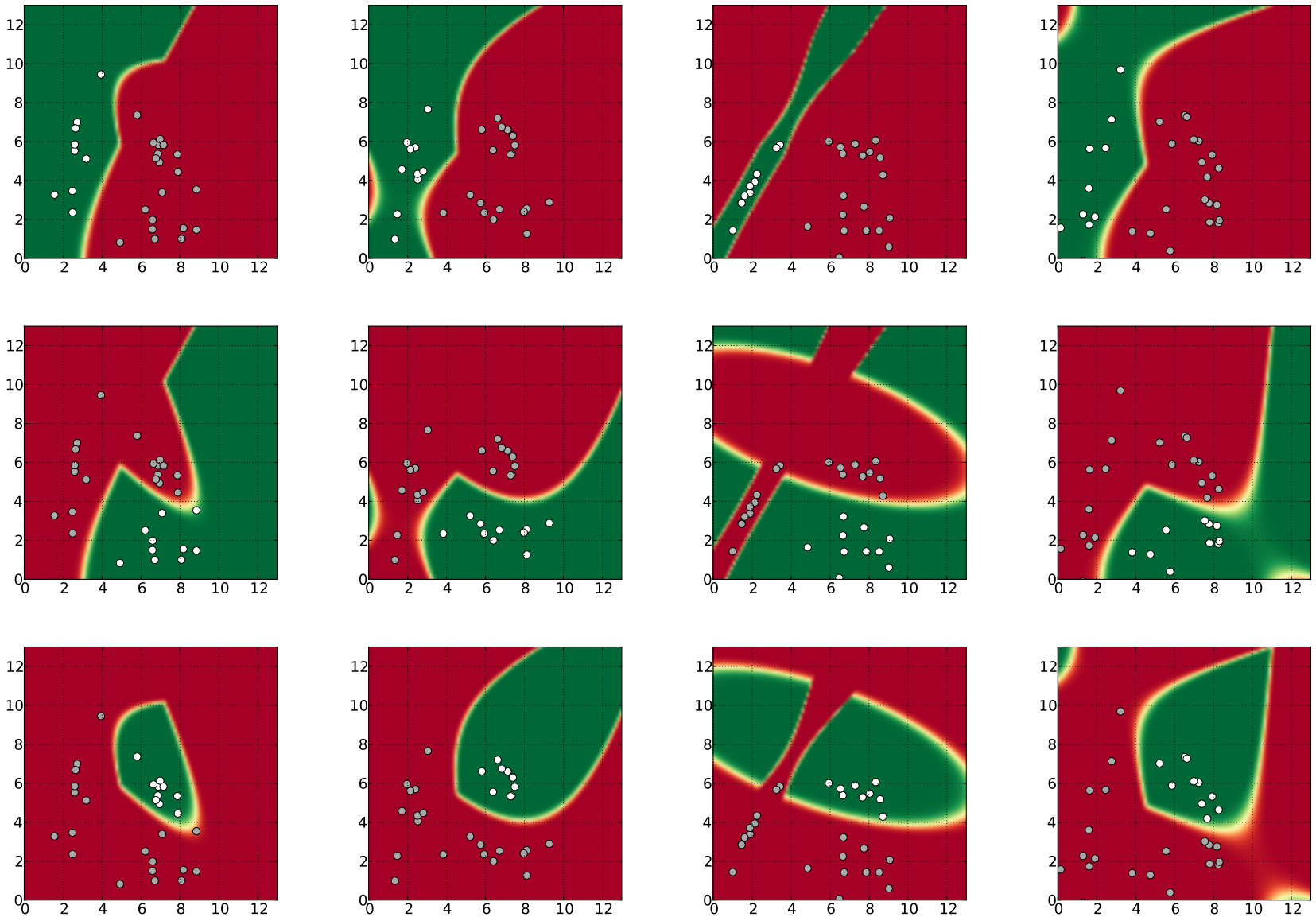


Run 2 *(10 samples per class)*

(CCDs left, posterior right)

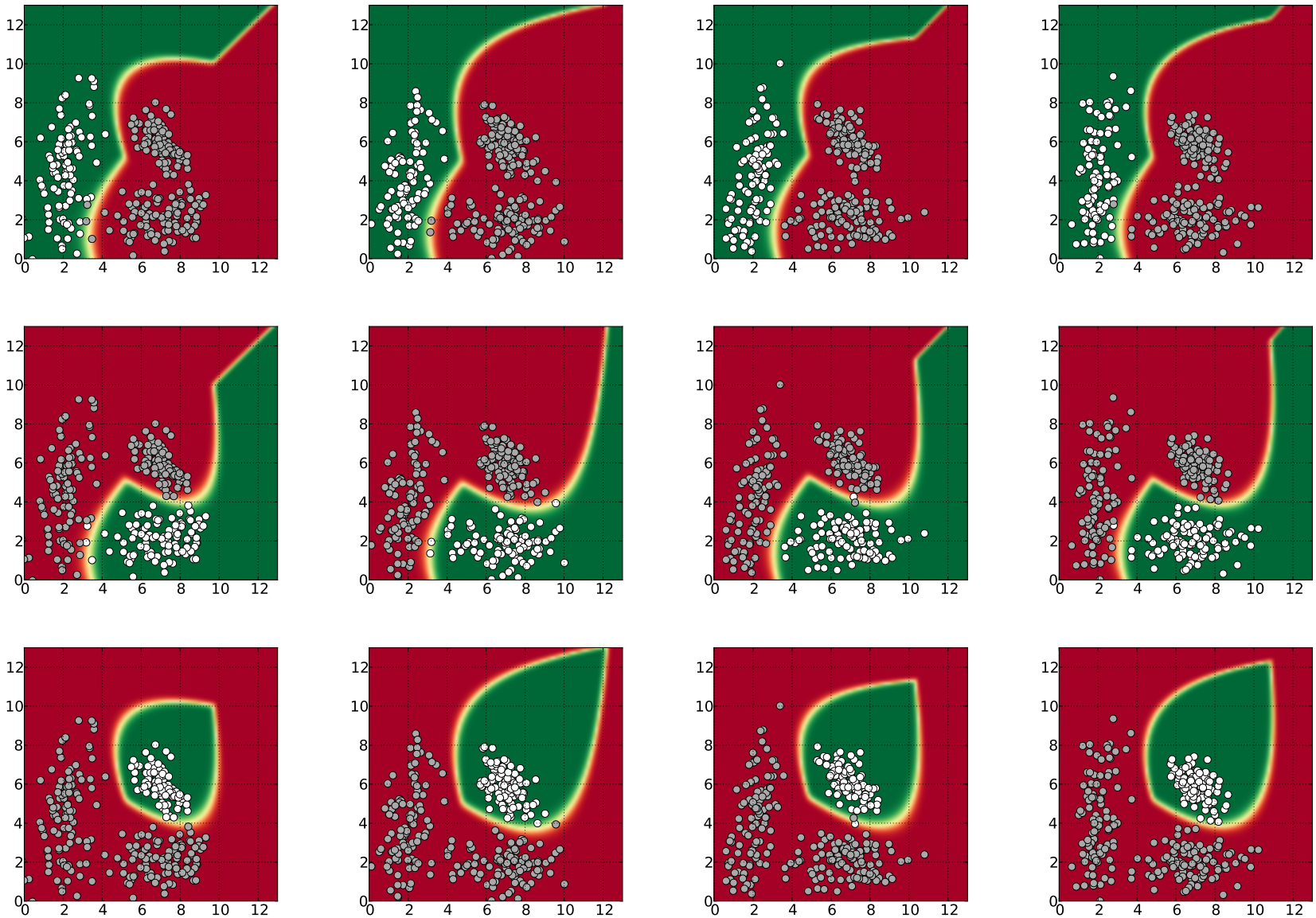
Overfitting in Naive Bayes

Posterior over 4 Runs *(class 0 (top), class 1 (center), class 2 (bottom))*



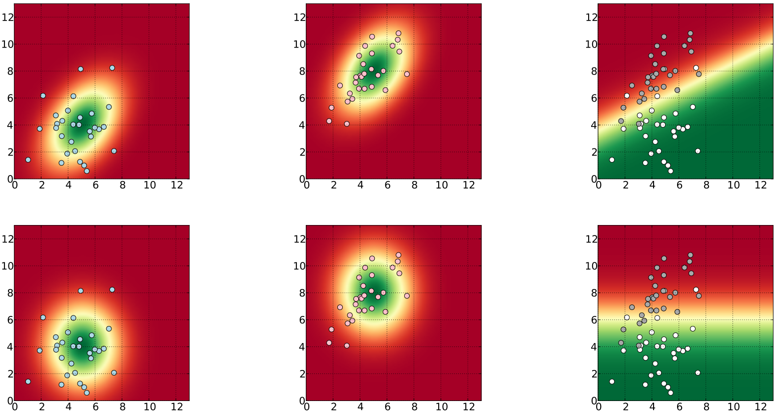
Overfitting (First Encounter)

Same experiment, but **100 samples** per class



Naive Bayes: Discussion

Naive Bayes: Discussion (cont'd)

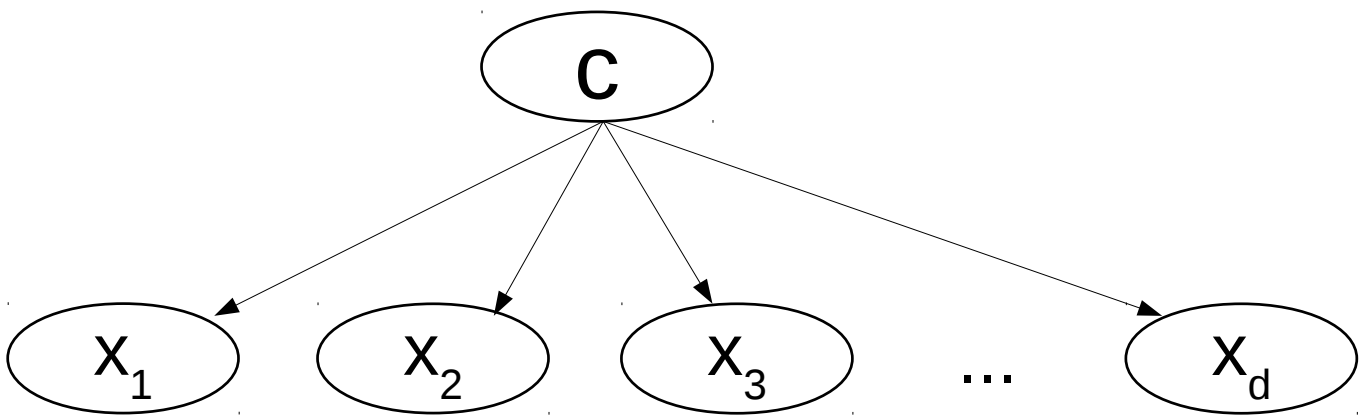


Outline

Naive Bayes: Graphical Illustration

- Generative models use the joint distribution $P(\mathbf{x}, c)$ of features $\mathbf{x} = (x_1, ..., x_d)$ and classs c.
- Naive Bayes takes the simplest approach possible *(all features are independent (given the class)!)*

$$P(x_1, ..., x_d|c) = P(x_1|c) \cdot P(x_2|c) \cdot ... \cdot P(x_d|c)$$



- Problem: We loose interdependencies between variables:

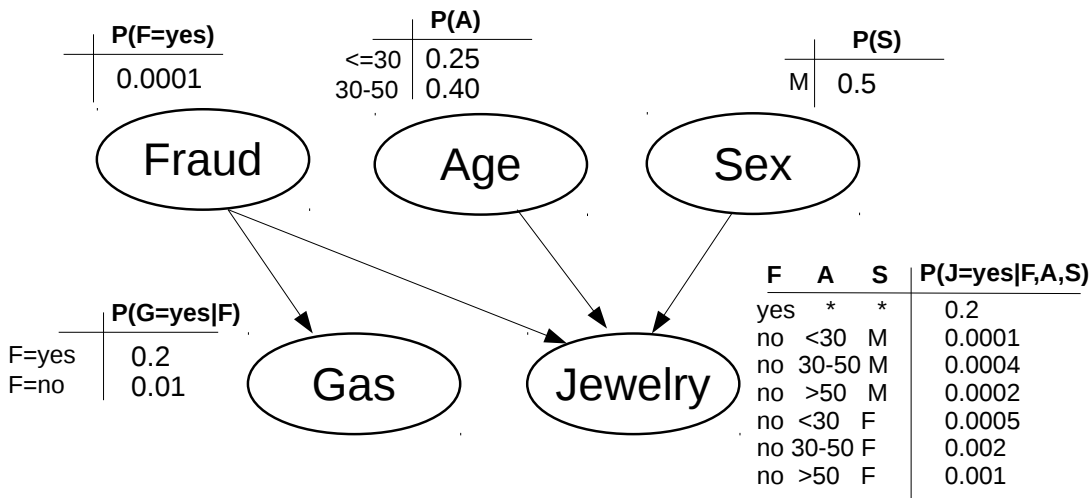
$$P(''six''|sports) \cdot P(''pack''|sports) \neq P(''six'',''pack''|sports)$$

Graphical Models

- Naive Bayes is an example of a **graphical model**. Such models capture the dependency structure between random variables.
- There are different flavors *(MRFs, CRFs, factor graphs, ...)*.

Here: Bayesian Networks

- **Example**[3]: credit card fraud detection
- `Gas/Jewelry`: was gas/jewelry bought in the last 24 hours?
- `Age/Sex`: age and sex of card holder



[3] from David Heckerman: "A Tutorial on Learning With Bayesian Networks", 1995.

Graphical Models

Definition (Bayesian Network)

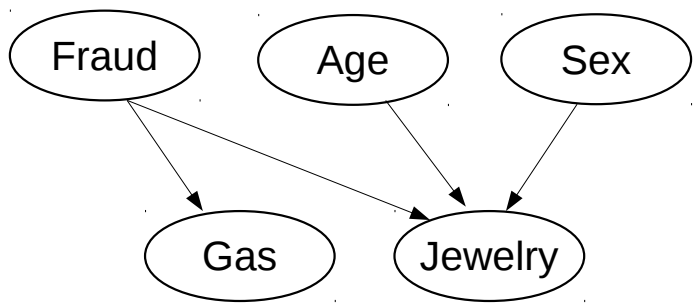
Given a set of random variables $\mathbf{X} = \{X_1, ..., X_d\}$*, a Bayesian network is a* **directed acyclic graph** *(DAG) with* $\mathbf{X}$ *as nodes. Let* $\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_d)$ *be a realization of* $(X_1, ..., X_d)$*, and* $\mathbf{pa}(\mathbf{x_i})$ *be a realization of* X_i*'s parents. Then the joint distribution of all variables is given by:*

$$P(\mathbf{x}) = \prod_{i=1}^{d} P(\mathbf{x}_i \,|\, \mathbf{pa}(\mathbf{x}_i))$$

Remarks

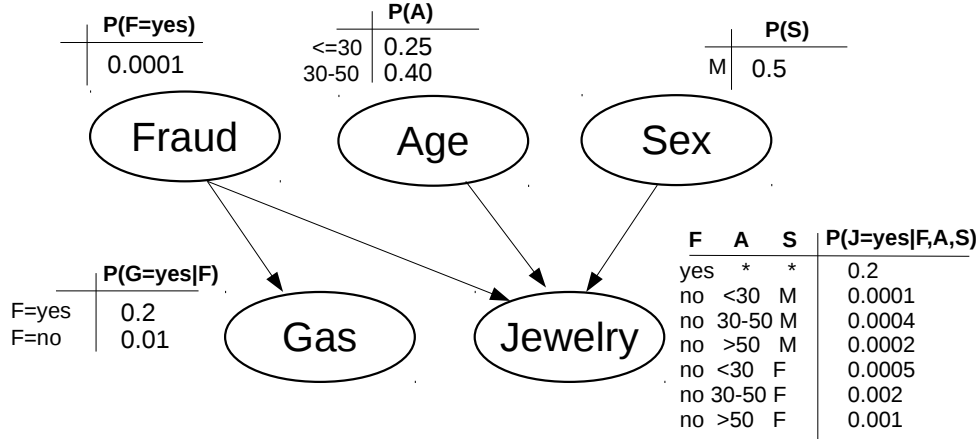
- In the example:

$$P(f, a, s, g, j) = P(f) \cdot P(a) \cdot P(s) \cdot P(g|f) \cdot P(j|f, a, s)$$



Bayesian Networks: Inference

Building a Fraud Detector



$$
\begin{aligned}
&P(f|a,s,g,j) \\
&= P(f,a,s,g,j)/P(a,s,g,j) \\
&= P(f,a,s,g,j)/\sum_{f'} P(f',a,s,g,j) \\
&= \frac{P(f)\cdot P(a)\cdot P(s)\cdot P(g|f)\cdot P(j|f,a,s)}{\sum_{f'} P(f')\cdot P(a)\cdot P(s)\cdot P(g|f)\cdot P(j|f',a,s)}
\end{aligned}
$$

Example

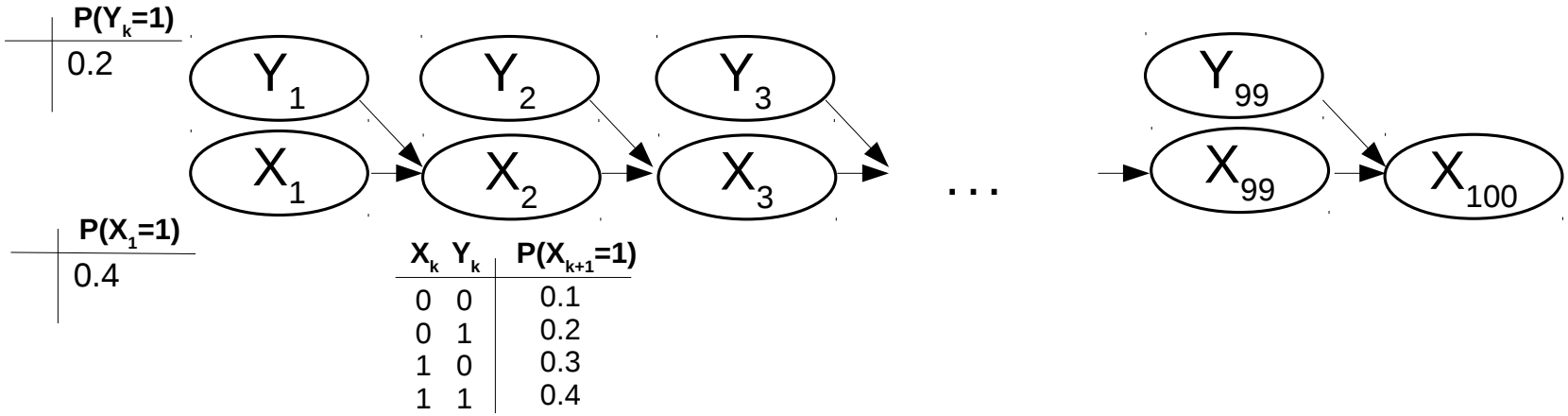
What is the probability of a fraud, given a < 30 male card owner who also purchased gas and jewelry?

$$P(F = yes, A \leq 30, S = M, G = Y, J = Y) = 0.0001 \cdot 0.25 \cdot 0.5 \cdot 0.2 \cdot 0.2 = 5 \cdot 10^{-7}$$

$$P(F = no,\ A \leq 30, S = M, G = Y, J = Y) = 0.9999 \cdot 0.25 \cdot 0.5 \cdot 0.01 \cdot 0.0001 = 1.25 \cdot 10^{-7}$$

$$P(F = yes|a,s,g,j) = \frac{5 \cdot \cancel{10^{-7}}}{1.25 \cdot \cancel{10^{-7}} + 5 \cdot \cancel{10^{-7}}} = 80\%$$

Bayesian Networks: Inference (Naive)



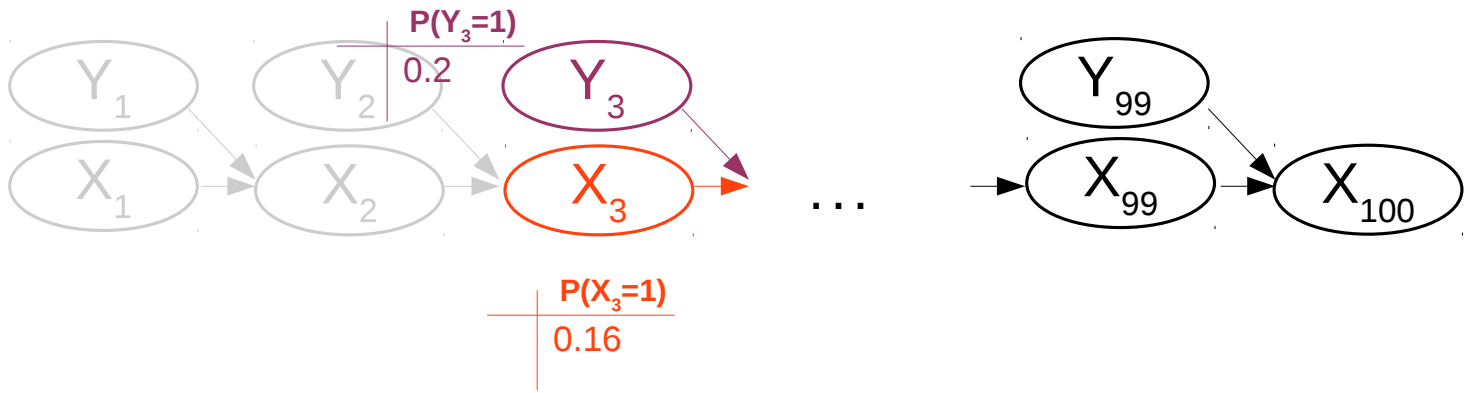
Example

- Compute $P(X_{100} = 1) \rightarrow 4^{99}$ combinations to try
- **Solution**: Compute $P(X_2 = 1)$, then $P(X_3 = 1)$, ...

$$P(X_2 = 1) = \sum_{x_1, y_1 = 0}^{1} P(x_1) \cdot P(y_1) \cdot P(X_2 = 1 | x_1, y_1) = 0.2$$

$$P(X_3 = 1) = \sum_{x_2, y_2 = 0}^{1} P(x_2) \cdot P(y_2) \cdot P(X_3 = 1 | x_2, y_2) = 0.16$$

Bayesian Networks: Inference



Solution: Belief Propagation

- We compute **local** probability distributions ("messages") and pass them along the edges of the graph
- This approach is also known as **message passing**
- Effort in the example: $4 \cdot 99$
- Remark: This can become more complex in case of (undirected) circles in the graph.

Bayesian Networks: Discussion

What are the Benefits?[4]

1. "Bayesian networks handle incomplete data"
   - age unknown? $\rightarrow$ simply marginalize over latent variables: $P(f|g,s,j) = \sum_{a'} P(f,a',s,j,g) / \sum_{f',a'} P(f',a',s,j,g)$
2. "Bayesian networks model causal relationships"
   - We can learn the distributions and structure of the network!
   - Analysts can draw insight from this structure *(in which user segment does an ad improve sales?)*
3. "Bayesian networks combine domain knowledge and data"
   - we can learn some parts of the network while defining others manually

Drawbacks?

- Inference becomes intracktable quickly, especially for high-dimensional problems
- Same for learning *(we don't deal with this here)*.

[4] from David Heckerman: "A Tutorial on Learning With Bayesian Networks", 1995.

References

[1] Scikit-Learn Landing Page.
`http://scikit-learn.org` (retrieved: Oct 2016).