Algorithmisches Lernen/Machine Learning

Part 1: Stefan Wermter

- Introduction
- Connectionist Learning (e.g. Neural Networks)
- Decision-Trees, Genetic Algorithms

Part 2: Norman Hendrich

- Support-Vector Machines
- Learning of Symbolic Structures
- **Bayesian Learning**
- Dimensionality Reduction

Part 3: Jianwei Zhang

- Function approximation
- Reinforcement Learning
- Applications in Robotics

Bayesian Learning

- Bayesian Reasoning
- Bayes Optimal Classifier
- Naïve Bayes Classifier
- Cost-Sensitive Decisions
- Modelling with Probability Density Functions
- Parameter Estimation
- Bayesian Networks
- Markov Models
- Dynamic Bayesian Networks
- Conditional Random Fields

Bayesian Learning

- Bayesian Reasoning
- Bayes Optimal Classifier
- Naïve Bayes Classifier
- Cost-Sensitive Decisions
- Modelling with Probability Density Functions
- Parameter Estimation
- Bayesian Networks
- Markov Models
- Dynamic Bayesian Networks
- Conditional Random Fields

Bayesian Reasoning

- derive the probability of a hypothesis h about some observation $\vec{x}$
- a priori probability: probability of the hypothesis prior to the observation $P(h)$
- a posteriori probability: probability of the hypothesis after observation $P(h|\vec{x})$
- observation can have discrete or continuous values
- continuous values: probability density functions $p(h|\vec{x})$ instead of probabilities
- error optimal decision: choose the hypothesis which maximizes the a posteriori probability (MAP-decision)

Bayesian Reasoning

- a posteriori probability is difficult to estimate
- Bayes' rule provides the missing link

$$P(h, \vec{x}) = P(\vec{x}, h) = P(h)P(\vec{x}|h) = P(\vec{x})P(h|\vec{x})$$

$$P(h|\vec{x}) = \frac{P(h)P(\vec{x}|h)}{P(\vec{x})}$$

Bayesian Reasoning

- classification: using the posterior probability as a target function

$$h_{MAP} = \arg\max_{h_i \in H} \frac{P(h_i)P(\vec{x}|h_i)}{P(\vec{x})} = \arg\max_{h_i \in H} P(h_i)P(\vec{x}|h_i)$$

- simplified form: maximum likelihood decision (e.g. if the priors are uniform)

$$h_{MAP} = \arg\max_{h_i \in H} P(\vec{x}|h)$$

Bayesian Reasoning

- allows
  - to include domain knowledge (prior probabilities)
  - to deal with inconsistent training data
  - to provide probabilistic results (confidence)
- but: probability distributions have to be estimated
  $\rightarrow$ usually many parameters

Bayesian Reasoning

- derived results: Bayesian analysis of learning paradigms may uncover their hidden assumptions, even if they are not probabilistic:
  - Every consistent learner outputs a MAP hypothesis under the assumption of uniform prior probabilities for all hypotheses and deterministic, noise-free training data
  - If the real training data can be assumed to be produced out of ideal ones by adding a normal-distributed noise term, any learner that minimizes the mean-squared error yields a ML hypothesis
  - If an observed Boolean value is a probabilistic function of the input value, minimizing cross entropy in neural networks yields a ML hypothesis

Bayesian Reasoning

- derived results (cont.):
  - If optimal encodings for the hypotheses and the training data given the hypothesis are chosen, selecting the hypothesis according to the principle of minimal description length gives a MAP hypothesis

$$h_{MDL} = \arg\min_{h \in H} L_{C_1}(h) + L_{C_2}(D|h)$$

Bayesian Learning

- Bayesian Reasoning
- Bayes Optimal Classifier
- Naïve Bayes Classifier
- Modelling with Probability Density Functions
- Parameter Estimation
- Bayesian Networks
- Markov Models
- Dynamic Bayesian Networks
- Conditional Random Fields

Bayes Optimal Classifier

- Bayes classifier does not always produce a true MAP decision
- e.g. for composite results

hypothesis	h_1	h_2	h_3
posterior probability	0.3	0.4	0.3

- maximum of posteriors gives h_2
- but if a new observation is classified positive by h_2 but negative by h_1 and h_3 the MAP decision would be "negative"
- extension of the Bayes classifier to composite decisions separating hypotheses h from decisions v

$$v_{MAP} = \arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|\vec{x})$$

- simplification for $P(v|h) \in \{0, 1\}$

Bayesian Learning

- Bayesian Reasoning
- Bayes Optimal Classifier
- Naïve Bayes Classifier
- Cost-Sensitive Decisions
- Modelling with Probability Density Functions
- Parameter Estimation
- Bayesian Networks
- Markov Models
- Dynamic Bayesian Networks
- Conditional Random Fields

Naïve Bayes Classifier

- Bayes Optimal Classifier is too expensive

$$v_{MAP} = \arg\max_{v_j \in V} P(v_j|\vec{x}) = \arg\max_{v_j \in V} P(v_j|x_1, x_2, ..., x_n)$$
$$= \arg\max_{v_j \in V} P(v_j)P(x_1, x_2, ..., x_n|v_j)$$

- prohibitively many parameter to estimate
- independence assumption:

$P(x_i|v_j)$ is independent of $P(x_k|v_j)$ for $i \neq k$

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(x_i|v_j)$$

- simple training
- usually good results

Bayesian Learning

- Bayesian Reasoning
- Bayes Optimal Classifier
- Naïve Bayes Classifier
- Cost-Sensitive Decisions
- Modelling with Probability Density Functions
- Parameter Estimation
- Bayesian Networks
- Markov Models
- Dynamic Bayesian Networks
- Conditional Random Fields

Cost-Sensitive Decisions

- error optimal classification not always welcome: highly asymmetric distributions
- diseases, errors, failures, ...
- priors determine the decision
- including a cost function into the decision rule
- c_{ij} cost of predicting i when the true class is j
- cost matrix

$$C = \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots \\ c_{n1} & c_{n2} & \dots & c_{nn} \end{pmatrix}$$

Cost-Sensitive Decisions

- Bayes classifier with cost function can help to reduce false positives/negatives

$$h(\vec{x}) = \arg\min_{h_i \in H} \sum_j c_{ij} \; p(h_j|\vec{x})$$

- alternative: biased sampling of training data
  - not really effective

Cost-Sensitive Decisions

- not every cost matrix is a reasonable one
  $\rightarrow$ reasonableness conditions
  - correct decisions should be less expensive than incorrect ones
    $c_{ii} < c_{ij} \quad i \neq j$
  - a row in the cost matrix should not dominate another one
    - row m dominates row n: $\forall j . c_{mj} \geq c_{nj}$
    - optimal policy: always decide for the dominated class
- e.g. asymmetric cost function for diseases:

	actually not ill	actually ill
predict not ill	0	1
predict ill	9	0

Cost-Sensitive Decisions

- any two-class cost matrix can be changed by
  - adding a constant to every entry (shifting)
  - multiplying every entry with a constant (scaling)

  without affecting the optimal decision

$$\begin{pmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{pmatrix} \quad \Longrightarrow \quad \begin{pmatrix} 0 & c_{01} - c_{00} / c_{10} - c_{00} \\ 1 & c_{11} - c_{00} / c_{10} - c_{00} \end{pmatrix}$$

$\rightarrow$ actually only one degree of freedom!

Cost-Sensitive Decisions

- optimal decision require the expected cost of the decision to be larger than the expected cost for the alternative decisions
  e.g. two-class case

$$P(\oplus|x)\ c_{10} + P(\ominus|x)\ c_{11} \leq P(\oplus|x)\ c_{00} + P(\ominus|x)\ c_{01}$$

$$(1 - P(\ominus|x)\ c_{10} + P(\ominus|x)\ c_{11} \leq (1 - P(\ominus|x)\ c_{00} + P(\ominus|x)\ c_{01}$$

- threshold for making optimal cost-sensitive decisions

$$(1 - p^*)\ c_{10} + p^*\ c_{11} = (1 - p^*)\ c_{00} + p^*\ c_{01}$$

$$p^* = \frac{c_{10} - c_{00}}{c_{10} - c_{00} + c_{01} - c_{11}}$$

  can be used e.g. in decision tree learning

Cost-Sensitive Decisions

- costs are a dangerous perspective for many applications
  - e.g. rejecting a ”good” bank loan application is a missed opportunity not an actual loss
    $\rightarrow$ cost are easily measured against different baselines
  - benefits provides a more natural (uniform) baseline: cash flow
- costs/benfits are usually not constant for every instance
  - e.g. potential benefit/loss of a defaulted bank loan varies with the amount

Bayesian Learning

- Bayesian Reasoning
- Bayes Optimal Classifier
- Naïve Bayes Classifier
- Cost-Sensitive Decisions
- Modelling with Probability Density Functions
- Parameter Estimation
- Bayesian Networks
- Markov Models
- Conditional Random Fields

Modelling with Probability Density Functions

- probability density functions $p(\vec{x}|v_j)$ instead of $P(\vec{x}|v_j)$
- $P(\vec{x}|v_j)$ is always zero in a continuous domain
- choosing a distribution class, e.g. Gaussian or Laplace

$$p(x|v) = \mathcal{N}[x, \mu, \sigma] = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

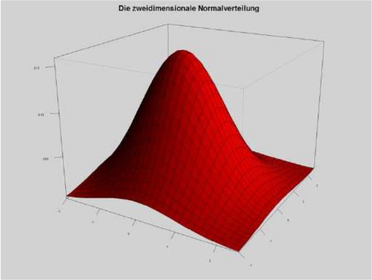
$$p(x|v) = \mathcal{L}[x, \mu, \sigma] = \frac{1}{2\sigma} e^{-\frac{|x-\mu|}{\sigma}}$$

- parameters: mean μ, variance σ

Modelling with Probability Density Functions

- distributions for multidimensional observations
- e.g. multivariate normal distribution

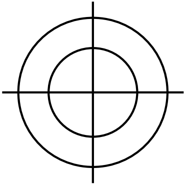
$$p(\vec{x}|v) = \mathcal{N}[\vec{x}, \vec{\mu}, \Sigma]$$



- parameters
  - vector of means $\vec{\mu}$
  - co-variance matrix Σ

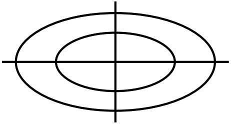
Modelling with Probability Density Functions

diagonal covariance matrix uniformly filled (rotation symmetry around the mean)



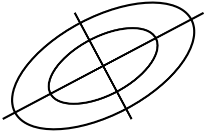
$$\sigma_{ij} = \begin{cases} n & \text{for } i = j \\ 0 & \text{else} \end{cases}$$

diagonal covariance matrix filled with arbitrary values (reflection symmetry)



$$\sigma_{ij} = \begin{cases} n_i & \text{for } i = j \\ 0 & \text{else} \end{cases}$$

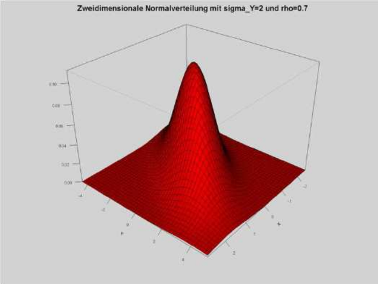
completely filled covariance matrix



Modelling with Probability Density Functions

- diagonal covariance matrix: uncorrelated features
  relativly small number of parameters to be trained
  → naïve Bayes classifier

- completely filled covariance matrix: correlated features
  high number of parameters to be trained



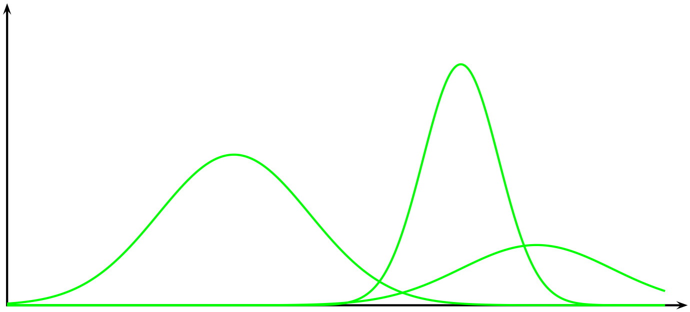
Modelling with Probability Density Functions

- decorrelation of the features: transformation of the feature space
  - Principal Component Analysis
  - Karhunen-Loève-Transformation

Modelling with Probability Density Functions

- compromise: mixture densities
- superposition of several Gaussians with uncorrelated features

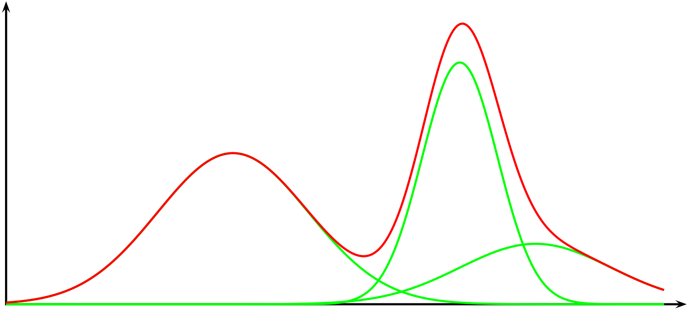
$$p(\vec{x}|v) = \sum_{m=1}^{M} c_m \, \mathcal{N}[\vec{x}, \vec{\mu}_m, \Sigma_m]$$



Modelling with Probability Density Functions

- compromise: mixture densities
- superposition of several Gaussians with uncorrelated features

$$p(\vec{x}|v) = \sum_{m=1}^{M} c_m \, \mathcal{N}[\vec{x}, \vec{\mu}_m, \Sigma_m]$$

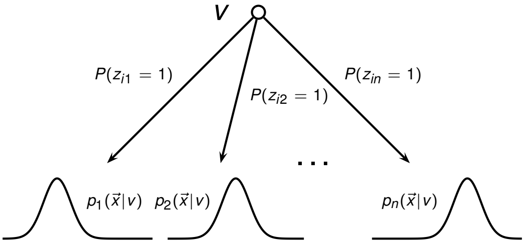


Modelling with Probability Density Functions

- mixture density functions introduce a hidden variable: Which Gaussian produced the value?
- two step stochastic process:
  - choosing a mixture randomly

$$z_{ij} = \begin{cases} 1 & \text{if } \vec{x}_i \text{ was generated by } p_j(\vec{x}|v) \\ 0 & \text{otherwise} \end{cases}$$

  - choosing a value randomly



- direct estimation of distribution parameters is not possible

Bayesian Learning

- Bayesian Reasoning
- Bayes Optimal Classifier
- Naïve Bayes Classifier
- Cost-Sensitive Decisions
- Modelling with Probability Density Functions
- Parameter Estimation
- Bayesian Networks
- Markov Models
- Dynamic Bayesian Networks
- Conditional Random Fields

Parameter estimation

- complete data
  - maximum likelihood estimation
  - Bayesian estimation
- incomplete data
  - expectation maximization
  - (gradient descent techniques)

Maximum Likelihood Estimation

- likelihood of the model M given the (training) data $\mathcal{D}$

$$L(M|\mathcal{D}) = \prod_{d \in \mathcal{D}} P(d|M)$$

- log-likelihood

$$LL(M|\mathcal{D}) = \prod_{d \in \mathcal{D}} log_2 P(d|M)$$

- choose among several possible models for describing the data according to the principle of maximum likelihood

$$\hat{\Theta} = \arg\max_{\Theta} L(M_\Theta|\mathcal{D}) = \arg\max_{\Theta} LL(M_\Theta|\mathcal{D})$$

- the models only differ in the set of parameters Θ

Maximum Likelihood Estimation

- complete data: estimating the parameters by counting

$$P(A = a) = \frac{N(A = a)}{\sum_{v \in dom(A)} N(A = v)}$$

$$P(A = a | B = b, C = c) = \frac{N(A = a, B = b, C = c)}{N(B = b, C = c)}$$

Bayesian Estimation

- sparse data bases result in pessimistic estimations for unseen events
  - if the count for an event in the data base is 0, the event ios considered impossible by the model
- Bayesian estimation: using an estimate of the prior probability as starting point for the counting
  - estimation of maximum a posteriori parameters
  - no zero counts can occur
  - if nothing else available use an even distribution as prior
  - Bayesian estimate in the binary case with an even distribution

$$P(yes) = \frac{n+1}{n+m+2}$$

    n: counts for yes, m: counts for no
  - effectively adding virtual counts to the estimate
  - alternative: smoothing as a post processing step

Incomplete Data

- missing at random:
  - probability that a value is missing depends only on the observed value
  - e.g. confirmation measurement: values are available only if the preceding measurement was positive/negative
- missing completely at random
  - probability that a value is missing is also independent of the value
  - e.g. stochastic failures of the measurement equipment
  - e.g. hidden/latent variables (mixture coefficients of a Gaussian mixture distribution)
- nonignorable:
  - neither MAR or MCAR
  - probability depends on the unseen values, e.g. exit polls for extremist parties

Expectation Maximization

Estimating the means of a Gaussian mixture distribution

- choose an initial hypothesis for $\Theta = (\mu_1, ..., \mu_k)$
- estimate the expected mean $E(z_{ij})$ given $\Theta = (\mu_1, ..., \mu_k)$
- recalculate the maximum likelihood estimate of the means: $\Theta' = (\mu'_1, ..., \mu'_k)$ assuming z_{ij}

$$z_{ij} = \begin{cases} 1 & \text{if } \vec{x}_i \text{ was generated by } p_j(\vec{x}|v) \\ 0 & \text{otherwise} \end{cases}$$

- replace μ_j by μ'_j and repeat until convergence

Expectation Maximization

- expectation:
  - ”complete” the data set using the current estimation $h = \Theta$ to calculate expectations for the missing values
  - applies the model to be learned (Bayesian inference)
- maximization:
  - use the ”completed” data set to find a new maximum likelihood estimation $h' = \Theta'$

Expectation Maximization

- generalizing the EM framework
- estimating the underlying distribution of not directly observable variables
- full data n+1-tuples $\langle \vec{x}_i, z_{i1}, ..., z_{ik} \rangle$
  only x_i can be observed
- training data: $X = \{\vec{x}_1, ..., \vec{x}_m\}$
- hidden information: $Z = \{z_1, ..., z_m\}$
- parameters of the distribution to be estimated: Θ
- Z can be treated as random variable with $p(Z) = f(\Theta, X)$
- full data: $Y = X \cup Z$
- hypothesis: h of Θ, needs to be revised into h'

Expectation Maximization

- goal of EM: $h' = \arg\max E(\log_2 p(Y|h'))$
- define a function $Q(h'|h) = E(\log_2 p(Y|h')|h, X)$

Estimation (E) step

Calculate $Q(h'|h)$ using the current hypothesis h and the observed data X to estimate the probability distribution over Y

$$Q(h'|h) \leftarrow E(\log_2 p(Y|h')|h, X)$$

Maximization (M) step

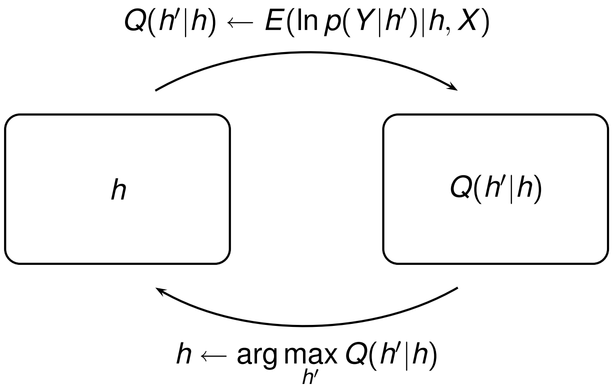
Replace hypothesis h by h' that maximizes the function Q

$$h \leftarrow \arg\max_{h'} Q(h'|h)$$

Expectation Maximization

- expectation step requires applying the model to be learned
  - Bayesian inference
- gradient ascent search
  - converges to the next local optimum
  - global optimum is not guaranteed

Expectation Maximization



- If Q is continuous, EM converges to the local maximum of the likelihood function $P(Y|h')$