Machine Learning - Lecture 18

Repetition

14.07.2015

Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de

leibe@vision.rwth-aachen.de

Announcements

- Today, I'll summarize the most important points from the lecture.
  - It is an opportunity for you to ask questions…
  - …or get additional explanations about certain topics.
  - *So, please do ask.*
- Today's slides are intended as an index for the lecture.
  - But they are not complete, won't be sufficient as only tool.
  - Also look at the exercises - they often explain algorithms in detail.

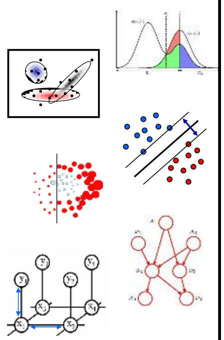
Announcements (2)

- Test exam on Thursday
  - During the regular lecture slot
  - Duration: 1h (instead of 2h as for the real exam)
  - *Purpose: prepare you for the questions you can expect*

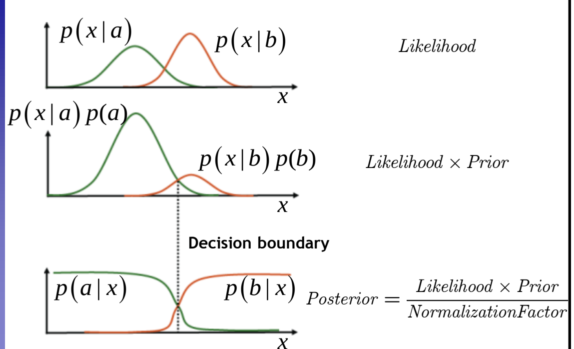
  - *All bonus points!*

Course Outline

- Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation
  - Mixture Models and EM
- Discriminative Approaches
  - Linear Discriminant Functions
  - Statistical Learning Theory & SVMs
  - Ensemble Methods & Boosting
  - Decision Trees & Randomized Trees
- Generative Models
  - Bayesian Networks
  - Markov Random Fields
  - Exact Inference



Recap: Bayes Decision Theory



Recap: Bayes Decision Theory

- Optimal decision rule
  - Decide for C_1 if

$$p(\mathcal{C}_1|x) > p(\mathcal{C}_2|x)$$

  - This is equivalent to

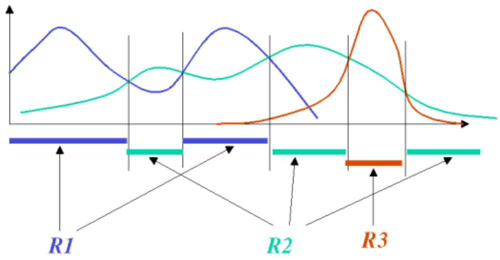
$$p(x|\mathcal{C}_1)p(\mathcal{C}_1) > p(x|\mathcal{C}_2)p(\mathcal{C}_2)$$

  - Which is again equivalent to (Likelihood-Ratio test)

$$\frac{p(x|\mathcal{C}_1)}{p(x|\mathcal{C}_2)} > \underbrace{\frac{p(\mathcal{C}_2)}{p(\mathcal{C}_1)}}_{\text{Decision threshold } \theta}$$

Recap: Bayes Decision Theory

- Decision regions: $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$, …



Slide credit: Bernt Schiele

Recap: Classifying with Loss Functions

- In general, we can formalize this by introducing a loss matrix L_{kj}

$$L_{kj} = loss\ for\ decision\ \mathcal{C}_j\ if\ truth\ is\ \mathcal{C}_k.$$

- Example: cancer diagnosis

$$L_{cancer\ diagnosis} = \begin{array}{c} \\ \text{cancer} \\ \text{normal} \end{array} \begin{array}{cc} \text{cancer} & \text{normal} \\ \left(\begin{array}{c} 0 \\ 1 \end{array} \right. & \left. \begin{array}{c} 1000 \\ 0 \end{array} \right) \end{array}$$

(Rows: Truth; columns: Decision)

Recap: Minimizing the Expected Loss

see Exercise 1.2

- Optimal solution minimizes the loss.
  - But: loss function depends on the true class, which is unknown.
- Solution: Minimize the expected loss

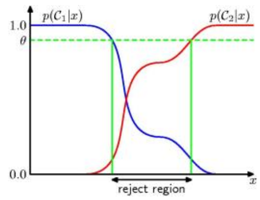
$$\mathbb{E}[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p(\mathbf{x}, \mathcal{C}_k)\, \mathrm{d}\mathbf{x}$$

- This can be done by choosing the regions $\mathcal{R}_j$ such that

$$\mathbb{E}[L] = \sum_k L_{kj} p(\mathcal{C}_k|\mathbf{x})$$

which is easy to do once we know the posterior class probabilities $p(\mathcal{C}_k|\mathbf{x})$.

Recap: The Reject Option

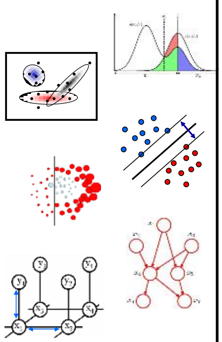


- Classification errors arise from regions where the largest posterior probability $p(\mathcal{C}_k|\mathbf{x})$ is significantly less than 1.
  - These are the regions where we are relatively uncertain about class membership.
  - For some applications, it may be better to reject the automatic decision entirely in such a case and e.g. consult a human expert.

Image source: C.M. Bishop, 2006

Course Outline

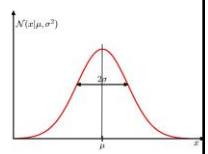
- Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation
  - Mixture Models and EM
- Discriminative Approaches
  - Linear Discriminant Functions
  - Statistical Learning Theory & SVMs
  - Ensemble Methods & Boosting
  - Decision Trees & Randomized Trees
- Generative Models
  - Bayesian Networks
  - Markov Random Fields
  - Exact Inference



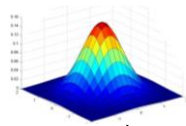
Recap: Gaussian (or Normal) Distribution

- One-dimensional case
  - Mean μ
  - Variance σ^2

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{ -\frac{(x-\mu)^2}{2\sigma^2} \right\}$$



- Multi-dimensional case
  - Mean μ
  - Covariance Σ



$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{ -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) \right\}$$

Image source: C.M. Bishop, 2006

Recap: Maximum Likelihood Approach

see Exercise 1.3

- Computation of the likelihood
  - Single data point: $p(x_n|\theta)$
  - Assumption: all data points $X = \{x_1, \ldots, x_n\}$ are independent

$$L(\theta) = p(X|\theta) = \prod_{n=1}^{N} p(x_n|\theta)$$

  - Log-likelihood

$$E(\theta) = -\ln L(\theta) = -\sum_{n=1}^{N} \ln p(x_n|\theta)$$

- Estimation of the parameters θ (Learning)
  - Maximize the likelihood (= minimize the negative log-likelihood)

⇒ Take the derivative and set it to zero.

$$\frac{\partial}{\partial\theta}E(\theta) = -\sum_{n=1}^{N} \frac{\frac{\partial}{\partial\theta}p(x_n|\theta)}{p(x_n|\theta)} \stackrel{!}{=} 0$$

Recap: Bayesian Learning Approach

- Bayesian view:
  - Consider the parameter vector θ as a random variable.
  - When estimating the parameters, what we compute is

$$p(x|X) = \int p(x, \theta|X) d\theta$$

Assumption: given θ, this doesn't depend on X anymore

$$p(x, \theta|X) = p(x|\theta, \cancel{X})p(\theta|X)$$

$$p(x|X) = \int \underbrace{p(x|\theta)}p(\theta|X)d\theta$$

This is entirely determined by the parameter θ (i.e. by the parametric form of the pdf).

Recap: Bayesian Learning Approach

- Discussion

Likelihood of the parametric form θ given the data set X.

Estimate for x based on parametric form θ

Prior for the parameters θ

$$p(x|X) = \int \frac{p(x|\theta)L(\theta)p(\theta)}{\underbrace{\int L(\theta)p(\theta)d\theta}} d\theta$$

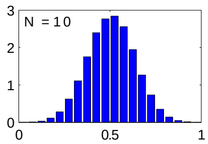
Normalization: integrate over all possible values of θ

  - The more uncertain we are about θ, the more we average over all possible parameter values.

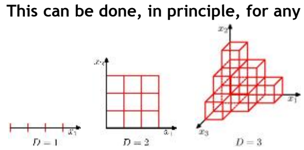
Recap: Histograms

- Basic idea:
  - Partition the data space into distinct bins with widths Δ_i and count the number of observations, n_i, in each bin.

$$p_i = \frac{n_i}{N\Delta_i}$$



  - Often, the same width is used for all bins, $\Delta_i = \Delta$.
  - This can be done, in principle, for any dimensionality D...



...but the required number of bins grows exponentially with D!

Recap: Kernel Density Estimation

see Exercise 1.4

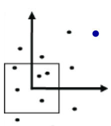
- Approximation formula:

$$p(\mathbf{x}) \approx \frac{K}{NV}$$

fixed V determine K → Kernel Methods

fixed K determine V → K-Nearest Neighbor

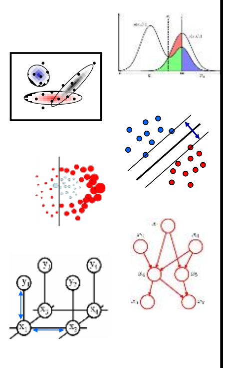
- Kernel methods
  - Place a *kernel window* k at location x and count how many data points fall inside it.



- K-Nearest Neighbor
  - Increase the volume V until the K next data points are found.

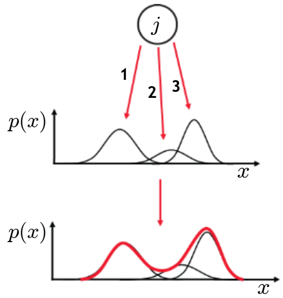
Course Outline

- Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation
  - Mixture Models and EM
- Discriminative Approaches
  - Linear Discriminant Functions
  - Statistical Learning Theory & SVMs
  - Ensemble Methods & Boosting
  - Decision Trees & Randomized Trees
- Generative Models
  - Bayesian Networks
  - Markov Random Fields
  - Exact Inference



Recap: Mixture of Gaussians (MoG)

- "Generative model"



$p(j) = \pi_j$ "Weight" of mixture component

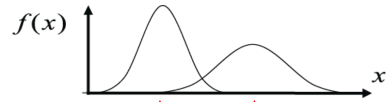
$p(x|\theta_j)$ Mixture component

Mixture density

$$p(x|\theta) = \sum_{j=1}^{M} p(x|\theta_j)p(j)$$

Recap: MoG - Iterative Strategy

- Assuming we knew the values of the hidden variable…



ML for Gaussian #1 ML for Gaussian #2

assumed known → 1 111 22 2 2 j

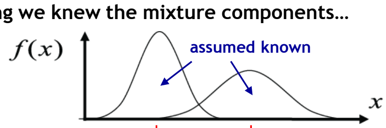
$h(j=1|x_n) =$ 1 111 00 0 0

$h(j=2|x_n) =$ 0 000 11 1 1

$$\mu_1 = \frac{\sum_{n=1}^{N} h(j=1|x_n)x_n}{\sum_{i=1}^{N} h(j=1|x_n)} \qquad \mu_2 = \frac{\sum_{n=1}^{N} h(j=2|x_n)x_n}{\sum_{i=1}^{N} h(j=2|x_n)}$$

Recap: MoG - Iterative Strategy

- Assuming we knew the mixture components…



$p(j=1|x)$ $p(j=2|x)$

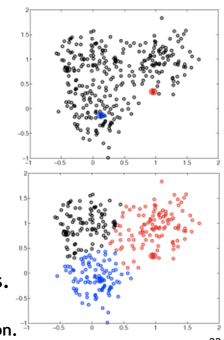
1 111 22 2 2 j

- Bayes decision rule: Decide $j = 1$ if

$$p(j=1|x_n) > p(j=2|x_n)$$

Recap: K-Means Clustering

- Iterative procedure
  1. Initialization: pick K arbitrary centroids (cluster means)
  2. Assign each sample to the closest centroid.
  3. Adjust the centroids to be the means of the samples assigned to them.
  4. Go to step 2 (until no change)
- Algorithm is guaranteed to converge after finite #iterations.
  - Local optimum
  - Final result depends on initialization.



Recap: EM Algorithm

see Exercise 1.5

- Expectation-Maximization (EM) Algorithm
  - E-Step: softly assign samples to mixture components

$$\gamma_j(\mathbf{x}_n) \leftarrow \frac{\pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{k=1}^{N} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \qquad \forall j = 1,\dots,K, \quad n = 1,\dots,N$$

  - M-Step: re-estimate the parameters (separately for each mixture component) based on the soft assignments

$$\hat{N}_j \leftarrow \sum_{n=1}^{N} \gamma_j(\mathbf{x}_n) \text{ = soft number of samples labeled } j$$

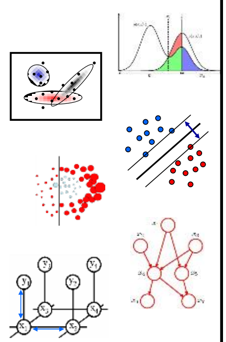
$$\hat{\pi}_j^{\text{new}} \leftarrow \frac{\hat{N}_j}{N}$$

$$\hat{\boldsymbol{\mu}}_j^{\text{new}} \leftarrow \frac{1}{\hat{N}_j} \sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)\mathbf{x}_n$$

$$\hat{\boldsymbol{\Sigma}}_j^{\text{new}} \leftarrow \frac{1}{\hat{N}_j} \sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_j^{\text{new}})(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_j^{\text{new}})^{\mathrm{T}}$$

Course Outline

- Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation
  - Mixture Models and EM
- Discriminative Approaches
  - Linear Discriminant Functions
  - Statistical Learning Theory & SVMs
  - Ensemble Methods & Boosting
  - Decision Trees & Randomized Trees
- Generative Models
  - Bayesian Networks
  - Markov Random Fields
  - Exact Inference

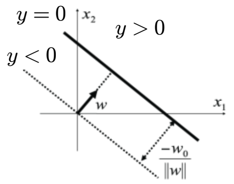


Recap: Linear Discriminant Functions

- Basic idea
  - Directly encode decision boundary
  - Minimize misclassification probability directly.
- Linear discriminant functions

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$$

weight vector — "bias" (= threshold)



  - $\mathbf{w}$, w_0 define a hyperplane in $\mathbb{R}^D$.
  - If a data set can be perfectly classified by a linear discriminant, then we call it linearly separable.

Recap: Least-Squares Classification

see Exercise 2.1

- Simplest approach
  - Directly try to minimize the sum-of-squares error

$$E(\mathbf{w}) = \sum_{n=1}^{N} (y(\mathbf{x}_n; \mathbf{w}) - \mathbf{t}_n)^2$$

$$E_D(\widetilde{\mathbf{W}}) = \frac{1}{2}\mathrm{Tr}\left\{(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})^{\mathrm{T}}(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})\right\}$$

  - Setting the derivative to zero yields

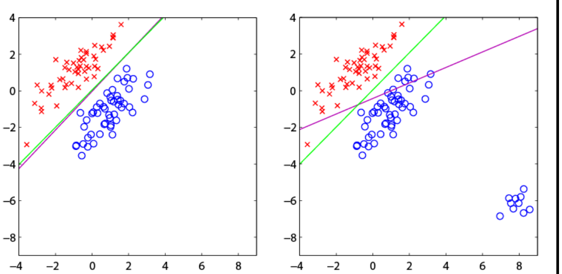
$$\widetilde{\mathbf{W}} = (\widetilde{\mathbf{X}}^{\mathrm{T}}\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}^{\mathrm{T}}\mathbf{T} = \widetilde{\mathbf{X}}^{\dagger}\mathbf{T}$$

  - We then obtain the discriminant function as

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^{\mathrm{T}}\widetilde{\mathbf{x}} = \mathbf{T}^{\mathrm{T}}\left(\widetilde{\mathbf{X}}^{\dagger}\right)^{\mathrm{T}}\widetilde{\mathbf{x}}$$

  - ⇒ Exact, closed-form solution for the discriminant function parameters.

Recap: Problems with Least Squares



- Least-squares is very sensitive to outliers!
  - The error function penalizes predictions that are "too correct".

Recap: Generalized Linear Models

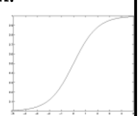
- Generalized linear model

$$y(\mathbf{x}) = g(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0)$$

  - $g(\cdot)$ is called an activation function and may be nonlinear.
  - The decision surfaces correspond to

$$y(\mathbf{x}) = const. \quad \Leftrightarrow \quad \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0 = const.$$

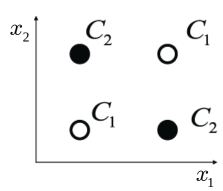
  - If g is monotonous (which is typically the case), the resulting decision boundaries are still linear functions of $\mathbf{x}$.
- Advantages of the non-linearity
  - Can be used to bound the influence of outliers and "too correct" data points.
  - When using a sigmoid for $g(\cdot)$, we can interpret the $y(\mathbf{x})$ as posterior probabilities.



$$g(a) \equiv \frac{1}{1 + \exp(-a)}$$

Recap: Linear Separability

- Up to now: restrictive assumption
  - Only consider linear decision boundaries
- Classical counterexample: XOR



Recap: Extension to Nonlinear Basis Fcts.

- Generalization
  - Transform vector $\mathbf{x}$ with M nonlinear basis functions $\phi_j(\mathbf{x})$:

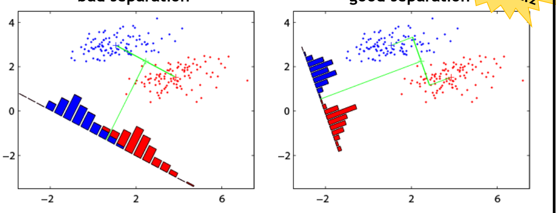
$$y_k(\mathbf{x}) = \sum_{j=1}^{M} w_{ki}\phi_j(\mathbf{x}) + w_{k0}$$

- Advantages
  - Transformation allows non-linear decision boundaries.
  - By choosing the right ϕ_j, every continuous function can (in principle) be approximated with arbitrary accuracy.
- Disadvantage
  - The error function can in general no longer be minimized in closed form.

⇒ Minimization with Gradient Descent

Recap: Classification as Dim. Reduction

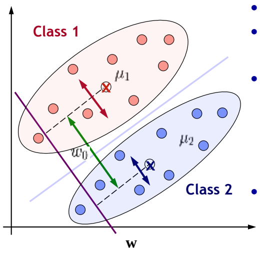
see Exercise 2.2



- Classification as dimensionality reduction
  - Interpret linear classification as a projection onto a lower-dim. space. $y = \mathbf{w}^{\mathrm{T}}\mathbf{x}$
  - ⇒ Learning problem: Try to find the projection vector w that maximizes class separation.

Image source: C.M. Bishop, 2006

Recap: Fisher's Linear Discriminant Analysis



- Maximize distance between classes
- Minimize distance within a class
- Criterion: $J(\mathbf{w}) = \frac{\mathbf{w}^{\mathrm{T}}\mathbf{S}_B\mathbf{w}}{\mathbf{w}^{\mathrm{T}}\mathbf{S}_W\mathbf{w}}$

  $\mathbf{S}_B$ ... between-class scatter matrix
  $\mathbf{S}_W$ ... within-class scatter matrix
- The optimal solution for w can be obtained as:
  $$\mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$$
- Classification function:
  $$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 \underset{\text{Class 2}}{\overset{\text{Class 1}}{\gtrless}} 0$$
  where $w_0 = -\mathbf{w}^T\mathbf{m}$

Slide adapted from Ales Leonardis

Recap: Probabilistic Discriminative Models

- Consider models of the form
$$p(\mathcal{C}_1|\phi) = y(\phi) = \sigma(\mathrm{w}^T\phi)$$
with
$$p(\mathcal{C}_2|\phi) = 1 - p(\mathcal{C}_1|\phi)$$
- This model is called logistic regression.
- Properties
  - Probabilistic interpretation
  - But discriminative method: only focus on decision hyperplane
  - Advantageous for high-dimensional spaces, requires less parameters than explicitly modeling $p(\phi|\mathcal{C}_k)$ and $p(\mathcal{C}_k)$.

Recap: Logistic Regression

- Let's consider a data set $\{\phi_n, t_n\}$ with $n = 1,\ldots,N$, where $\phi_n = \phi(\mathbf{x}_n)$ and $t_n \in \{0,1\}$, $\mathbf{t} = (t_1,\ldots,t_N)^T$.
- With $y_n = p(\mathcal{C}_1|\phi_n)$, we can write the likelihood as
$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}\{1-y_n\}^{1-t_n}$$
- Define the error function as the negative log-likelihood
$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}$$
  - This is the so-called cross-entropy error function.

Recap: Iterative Methods for Estimation

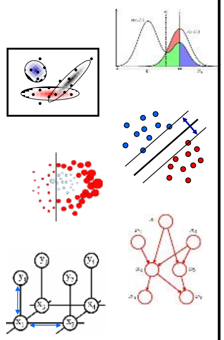
- Gradient Descent (1st order)
$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\,\nabla E(\mathbf{w})|_{\mathbf{w}^{(\tau)}}$$
  - Simple and general
  - Relatively slow to converge, has problems with some functions
- Newton-Raphson (2nd order)
$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\,\mathbf{H}^{-1}\nabla E(\mathbf{w})|_{\mathbf{w}^{(\tau)}}$$
  where $\mathbf{H} = \nabla\nabla E(\mathbf{w})$ is the Hessian matrix, i.e. the matrix of second derivatives.
  - Local quadratic approximation to the target function
  - Faster convergence

Recap: Iteratively Reweighted Least Squares

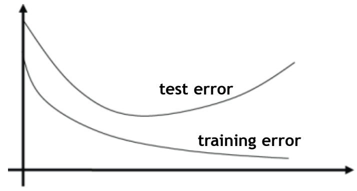
- Update equations
$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - (\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^T(\mathbf{y}-\mathbf{t})$$
$$= (\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi})^{-1}\left\{\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi}\mathbf{w}^{(\tau)} - \mathbf{\Phi}^T(\mathbf{y}-\mathbf{t})\right\}$$
$$= (\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{R}\mathbf{z}$$
with $\mathbf{z} = \mathbf{\Phi}\mathbf{w}^{(\tau)} - \mathbf{R}^{-1}(\mathbf{y}-\mathbf{t})$
- Very similar form to pseudo-inverse (normal equations)
  - But now with non-constant weighing matrix $\mathbf{R}$ (depends on $\mathbf{w}$).
  - Need to apply normal equations iteratively.
  - ⇒ Iteratively Reweighted Least-Squares (IRLS)

Course Outline

- **Fundamentals**
  - Bayes Decision Theory
  - Probability Density Estimation
  - Mixture Models and EM
- **Discriminative Approaches**
  - Linear Discriminant Functions
  - Statistical Learning Theory & SVMs
  - Ensemble Methods & Boosting
  - Decision Trees & Randomized Trees
- **Generative Models**
  - Bayesian Networks
  - Markov Random Fields
  - Exact Inference



Recap: Generalization and Overfitting



- **Goal: predict class labels of new observations**
  - Train classification model on limited training set.
  - The further we optimize the model parameters, the more the training error will decrease.
  - However, at some point the test error will go up again.
  - ⇒ *Overfitting to the training set!*

Recap: Risk

- **Empirical risk**
  - Measured on the training/validation set

$$R_{emp}(\alpha) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i;\alpha))$$

- **Actual risk** (= **Expected risk**)
  - Expectation of the error on *all* data.

$$R(\alpha) = \int L(y_i, f(\mathbf{x};\alpha))dP_{X,Y}(\mathbf{x}, y)$$

  - $P_{X,Y}(\mathbf{x}, y)$ is the probability distribution of $(\mathbf{x}, y)$. It is fixed, but typically unknown.
  - ⇒ In general, we can't compute the actual risk directly!

Recap: Statistical Learning Theory

- **Idea**
  - Compute an upper bound on the actual risk based on the empirical risk

$$R(\alpha) \cdot \quad R_{emp}(\alpha) + \epsilon(N, p^*, h)$$

  - where

    N: number of training examples

    p^*: probability that the bound is correct

    h: capacity of the learning machine ("VC-dimension")

Recap: VC Dimension

see Exercise 2.3

- **Vapnik-Chervonenkis dimension**
  - Measure for the capacity of a learning machine.
- **Formal definition:**
  - *If a given set of ℓ points can be labeled in all possible 2^ℓ ways, and for each labeling, a member of the set $\{f(\alpha)\}$ can be found which correctly assigns those labels, we say that the set of points is shattered by the set of functions.*
  - *The VC dimension for the set of functions $\{f(\alpha)\}$ is defined as the maximum number of training points that can be shattered by $\{f(\alpha)\}$.*

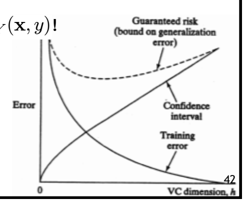
Recap: Upper Bound on the Risk

- **Important result (Vapnik 1979, 1995)**
  - With probability $(1-\eta)$, the following bound holds

$$R(\alpha) \cdot \quad R_{emp}(\alpha) + \underbrace{\sqrt{\frac{h(\log(2N/h)+1) - \log(\eta/4)}{N}}}_{\text{"VC confidence"}}$$

  - This bound is independent of $P_{X,Y}(\mathbf{x}, y)$!
  - If we know h (the VC dimension), we can easily compute the risk bound

$$R(\alpha) \cdot \quad R_{emp}(\alpha) + \epsilon(N, p^*, h)$$



Recap: Structural Risk Minimization

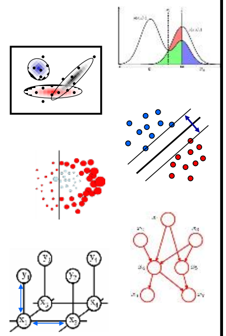
- How can we implement Structural Risk Minimization?

$$R(\alpha) \cdot \quad R_{emp}(\alpha) + \epsilon(N, p^*, h)$$

- Classic approach
  - Keep $\epsilon(N, p^*, h)$ constant and minimize $R_{emp}(\alpha)$.
  - $\epsilon(N, p^*, h)$ can be kept constant by controlling the model parameters.
- Support Vector Machines (SVMs)
  - Keep $R_{emp}(\alpha)$ constant and minimize $\epsilon(N, p^*, h)$.
  - In fact: $R_{emp}(\alpha) = 0$ for separable data.
  - Control $\epsilon(N, p^*, h)$ by adapting the VC dimension (controlling the "capacity" of the classifier).

Course Outline

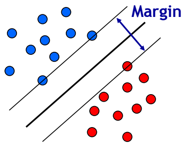
- Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation
  - Mixture Models and EM
- Discriminative Approaches
  - Linear Discriminant Functions
  - Statistical Learning Theory & SVMs
  - Ensemble Methods & Boosting
  - Decision Trees & Randomized Trees
- Generative Models
  - Bayesian Networks
  - Markov Random Fields
  - Exact Inference



Recap: Support Vector Machine (SVM)

- Basic idea
  - The SVM tries to find a classifier which maximizes the margin between pos. and neg. data points.
  - Up to now: consider linear classifiers

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = 0$$



- Formulation as a convex optimization problem
  - Find the hyperplane satisfying

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

under the constraints

$$t_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b) \geq 1 \quad \forall n$$

based on training data points $\mathbf{x}_n$ and target values $t_n \in \{-1, 1\}$.

Recap: SVM - Primal Formulation

- Lagrangian primal form

$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n \{t_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b) - 1\}$$

$$= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n \{t_n y(\mathbf{x}_n) - 1\}$$

- The solution of L_p needs to fulfill the KKT conditions
  - Necessary and sufficient conditions

$$\begin{aligned} a_n &\geq 0 \\ t_n y(\mathbf{x}_n) - 1 &\geq 0 \\ a_n\{t_n y(\mathbf{x}_n) - 1\} &= 0 \end{aligned}$$

KKT:

$$\begin{aligned} \lambda &\geq 0 \\ f(\mathbf{x}) &\geq 0 \\ \lambda f(\mathbf{x}) &= 0 \end{aligned}$$

Recap: SVM - Solution

- Solution for the hyperplane
  - Computed as a linear combination of the training examples

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \mathbf{x}_n$$

  - Sparse solution: $a_n \neq 0$ only for some points, the support vectors

⇒ Only the SVs actually influence the decision boundary!

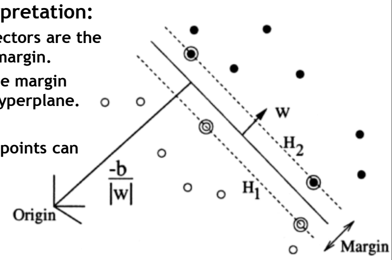
  - Compute b by averaging over all support vectors:

$$b = \frac{1}{N_{\mathcal{S}}} \sum_{n \in \mathcal{S}} \left(t_n - \sum_{m \in \mathcal{S}} a_m t_m \mathbf{x}_m^{\mathrm{T}} \mathbf{x}_n \right)$$

Recap: SVM - Support Vectors

- The training points for which $a_n > 0$ are called "support vectors".
- Graphical interpretation:
  - The support vectors are the points on the margin.
  - They *define* the margin and thus the hyperplane.

⇒ All other data points can be discarded!



Recap: SVM - Dual Formulation

see Exercise 2.4

- **Maximize**

$$L_d(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m (\mathbf{x}_m^{\mathrm{T}}\mathbf{x}_n)$$

under the conditions

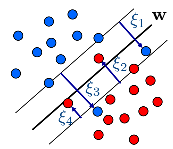
$$a_n \geq 0 \quad \forall n$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

- **Comparison**
  - L_d is equivalent to the primal form L_p, but only depends on a_n.
  - L_p scales with $\mathcal{O}(D^3)$.
  - L_d scales with $\mathcal{O}(N^3)$ - in practice between $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$.

Recap: SVM for Non-Separable Data

- **Slack variables**
  - One slack variable $\xi_n \geq 0$ for each training data point.
- **Interpretation**
  - $\xi_n = 0$ for points that are on the correct side of the margin.
  - $\xi_n = |t_n - y(\mathbf{x}_n)|$ for all other points.



Point on decision boundary: $\xi_n = 1$

Misclassified point: $\xi_n > 1$

  - We do not have to set the slack variables ourselves!

$\Rightarrow$ They are jointly optimized together with $\mathbf{w}$.

Recap: SVM - New Dual Formulation

see Exercise 2.5

- **New SVM Dual: Maximize**

$$L_d(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m (\mathbf{x}_m^{\mathrm{T}}\mathbf{x}_n)$$

under the conditions

$$0 \cdot a_n \cdot C$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

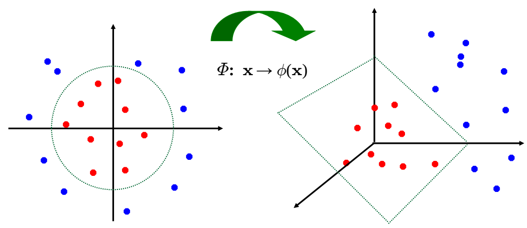
This is all that changed!

- **This is again a quadratic programming problem**

$\Rightarrow$ Solve as before...

Recap: Nonlinear SVMs

- **General idea: The original input space can be mapped to some higher-dimensional feature space where the training set is separable:**



Recap: The Kernel Trick

- **Important observation**
  - $\phi(\mathbf{x})$ only appears in the form of dot products $\phi(\mathbf{x})^{\mathrm{T}}\phi(\mathbf{y})$:

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + b$$

$$= \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n)^{\mathrm{T}}\phi(\mathbf{x}) + b$$

  - Define a so-called kernel function $k(\mathbf{x},\mathbf{y}) = \phi(\mathbf{x})^{\mathrm{T}}\phi(\mathbf{y})$.
  - Now, in place of the dot product, use the kernel instead:

$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}_n, \mathbf{x}) + b$$

  - The kernel function *implicitly* maps the data to the higher-dimensional space (without having to compute $\phi(\mathbf{x})$ explicitly)!

Recap: Kernels Fulfilling Mercer's Condition

- **Polynomial kernel**

$$k(\mathbf{x},\mathbf{y}) = (\mathbf{x}^{\mathrm{T}}\mathbf{y} + 1)^p$$

- **Radial Basis Function kernel**

$$k(\mathbf{x},\mathbf{y}) = \exp\left\{-\frac{(\mathbf{x}-\mathbf{y})^2}{2\sigma^2}\right\}$$

e.g. Gaussian

- **Hyperbolic tangent kernel**

$$k(\mathbf{x},\mathbf{y}) = \tanh(\kappa\mathbf{x}^{\mathrm{T}}\mathbf{y} + \delta)$$

e.g. Sigmoid

  - And many, many more, including kernels on graphs, strings, and symbolic data...

Recap: Kernels Fulfilling Mercer's Condition

- **Polynomial kernel**

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^{\mathrm{T}}\mathbf{y} + 1)^p$$

- **Radial Basis Function kernel**

$$k(\mathbf{x}, \mathbf{y}) = \exp\left\{-\frac{(\mathbf{x}-\mathbf{y})^2}{2\sigma^2}\right\}$$ e.g. Gaussian

- **Hyperbolic tangent kernel**

$$k(\mathbf{x}, \mathbf{y}) = \tanh(\kappa\mathbf{x}^{\mathrm{T}}\mathbf{y} + \delta)$$ (struck through) e.g. Sigmoid

Actually, that was wrong in the original SVM paper...

- And many, many more, including kernels on graphs, strings, and symbolic data…

Machine Learning, Summer '15

Slide credit: Bernt Schiele

Recap: Nonlinear SVM - Dual Formulation

see Exercise 2.5

- **SVM Dual: Maximize**

$$L_d(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_m, \mathbf{x}_n)$$

under the conditions

$$0 \cdot a_n \cdot C$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

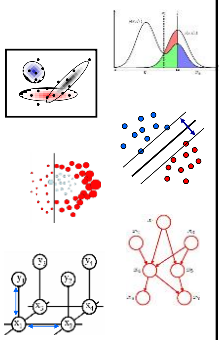
- **Classify new data points using**

$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}_n, \mathbf{x}) + b$$

Machine Learning, Summer '15

Course Outline

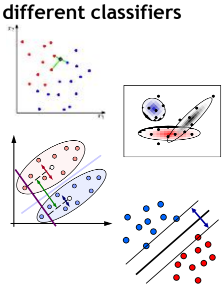
- **Fundamentals**
  - Bayes Decision Theory
  - Probability Density Estimation
  - Mixture Models and EM
- **Discriminative Approaches**
  - Linear Discriminant Functions
  - Statistical Learning Theory & SVMs
  - Ensemble Methods & Boosting
  - Decision Trees & Randomized Trees
- **Generative Models**
  - Bayesian Networks
  - Markov Random Fields
  - Exact Inference



Machine Learning, Summer '15

Recap: Classifier Combination

- **We've seen already a variety of different classifiers**
  - k-NN
  - Bayes classifiers
  - Fisher's Linear Discriminant
  - SVMs

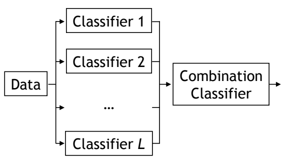


- **Each of them has their strengths and weaknesses…**
  - Can we improve performance by combining them?

Machine Learning, Summer '15

Recap: Stacking

- **Idea**
  - Learn L classifiers (based on the training data)
  - Find a meta-classifier that takes as input the output of the L first-level classifiers.



- **Example**
  - Learn L classifiers with leave-one-out.
  - Interpret the prediction of the L classifiers as L-dimensional feature vector.
  - Learn "level-2" classifier based on the examples generated this way.

Machine Learning, Summer '15

Slide credit: Bernt Schiele

Recap: Stacking

- **Why can this be useful?**
  - Simplicity
    - We may already have several existing classifiers available.
    - ⇒ No need to retrain those, they can just be combined with the rest.
  - Correlation between classifiers
    - The combination classifier can learn the correlation.
    - ⇒ Better results than simple Naïve Bayes combination.
  - Feature combination
    - E.g. combine information from different sensors or sources (vision, audio, acceleration, temperature, radar, etc.).
    - We can get good training data for each sensor individually, but data from all sensors together is rare.
    - ⇒ Train each of the L classifiers on its own input data. Only combination classifier needs to be trained on combined input.

Machine Learning, Summer '15

Recap: Bayesian Model Averaging

- Model Averaging
  - Suppose we have H different models $h = 1,\ldots,H$ with prior probabilities $p(h)$.
  - Construct the marginal distribution over the data set

$$p(\mathbf{X}) = \sum_{h=1}^{H} p(\mathbf{X}|h)p(h)$$

- Average error of committee

$$\mathbb{E}_{COM} = \frac{1}{M}\mathbb{E}_{AV}$$

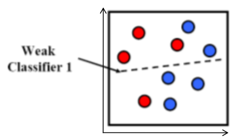
  - This suggests that the average error of a model can be reduced by a factor of M simply by averaging M versions of the model!
  - Unfortunately, this assumes that the errors are all uncorrelated. In practice, they will typically be highly correlated.

Recap: AdaBoost - "Adaptive Boosting"

- Main idea [Freund & Schapire, 1996]
  - Instead of resampling, reweight misclassified training examples.
    - Increase the chance of being selected in a sampled training set.
    - Or increase the misclassification cost when training on the full set.
- Components
  - $h_m(\mathbf{x})$: "weak" or base classifier
    - Condition: <50% training error over any distribution
  - $H(\mathbf{x})$: "strong" or final classifier
- AdaBoost:
  - Construct a strong classifier as a thresholded linear combination of the weighted weak classifiers:

$$H(\mathbf{x}) = sign\left(\sum_{m=1}^{M} \alpha_m h_m(\mathbf{x})\right)$$

Recap: AdaBoost - Intuition

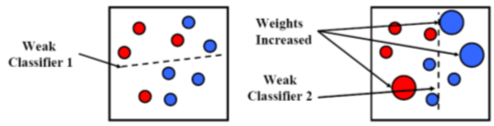


Consider a 2D feature space with positive and negative examples.

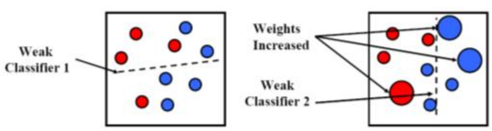
Each weak classifier splits the training examples with at least 50% accuracy.

Examples misclassified by a previous weak learner are given more emphasis at future rounds.

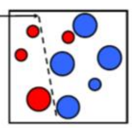
Recap: AdaBoost - Intuition



Recap: AdaBoost - Intuition



Weak classifier 3



Final classifier is combination of the weak classifiers

Recap: AdaBoost - Algorithm

see Exercise 3.1

1. Initialization: Set $w_n^{(1)} = \frac{1}{N}$ for $n = 1,\ldots,N$.
2. For $m = 1,\ldots,M$ iterations
   a) Train a new weak classifier $h_m(\mathbf{x})$ using the current weighting coefficients $\mathbf{W}^{(m)}$ by minimizing the weighted error function

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(h_m(\mathbf{x}) \neq t_n) \qquad I(A) = \begin{cases} 1, & \text{if } A \text{ is true} \\ 0, & \text{else} \end{cases}$$

   b) Estimate the weighted error of this classifier on $\mathbf{X}$:

$$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(h_m(\mathbf{x}) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}$$

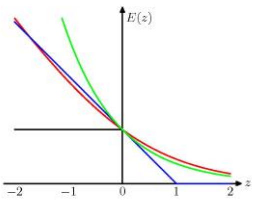
   c) Calculate a weighting coefficient for $h_m(\mathbf{x})$:

$$\alpha_m = \ln\left\{\frac{1-\epsilon_m}{\epsilon_m}\right\}$$

   d) Update the weighting coefficients:

$$w_n^{(m+1)} = w_n^{(m)} \exp\left\{\alpha_m I(h_m(\mathbf{x}_n) \neq t_n)\right\}$$

Recap: Comparing Error Functions

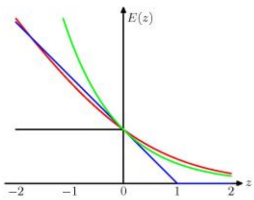


- Ideal misclassification error function
- "Hinge error" used in SVMs
- Exponential error function
  - Continuous approximation to ideal misclassification function.
  - Sequential minimization leads to simple AdaBoost scheme.
  - Disadvantage: exponential penalty for large negative values!

  ⇒ Less robust to outliers or misclassified data points!

Image source: Bishop, 2006

Recap: Comparing Error Functions



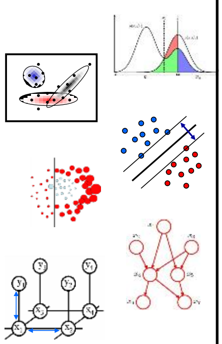
- Ideal misclassification error function
- "Hinge error" used in SVMs
- Exponential error function
- "Cross-entropy error" $E = -\sum \{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}$
  - Similar to exponential error for z>0.
  - Only grows linearly with large negative values of z.

  ⇒ Make AdaBoost more robust by switching ⇒ "GentleBoost"

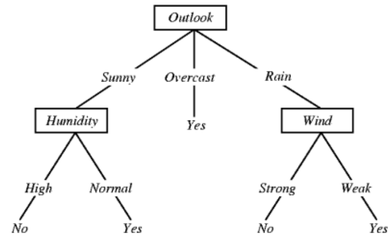
Image source: Bishop, 2006

Course Outline

- Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation
  - Mixture Models and EM
- Discriminative Approaches
  - Linear Discriminant Functions
  - Statistical Learning Theory & SVMs
  - Ensemble Methods & Boosting
  - Decision Trees & Randomized Trees
- Generative Models
  - Bayesian Networks
  - Markov Random Fields
  - Exact Inference



Recap: Decision Trees



- Example:
  - "Classify Saturday mornings according to whether they're suitable for playing tennis."

Image source: T. Mitchell, 1997

Recap: CART Framework

- Six general questions
  1. Binary or multi-valued problem?
     - I.e. how many splits should there be at each node?
  2. Which property should be tested at a node?
     - I.e. how to select the query attribute?
  3. When should a node be declared a leaf?
     - I.e. when to stop growing the tree?
  4. How can a grown tree be simplified or pruned?
     - Goal: reduce overfitting.
  5. How to deal with impure nodes?
     - I.e. when the data itself is ambiguous.
  6. How should missing attributes be handled?

Recap: Picking a Good Splitting Feature

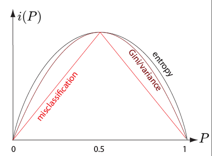
see Exercise 3.2

- Goal
  - Select the query (=split) that decreases impurity the most

$$\triangle i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R)$$

- Impurity measures
  - Entropy impurity (information gain):

$$i(N) = -\sum_j p(\mathcal{C}_j|N)\log_2 p(\mathcal{C}_j|N)$$



  - Gini impurity:

$$i(N) = \sum_{i \neq j} p(\mathcal{C}_i|N)p(\mathcal{C}_j|N) = \frac{1}{2}\left[1 - \sum_j p^2(\mathcal{C}_j|N)\right]$$

Image source: R.O. Duda, P.E. Hart, D.G. Stork, 2001

Recap: Computational Complexity

- Given
  - Data points $\{\mathbf{x}_1,\ldots,\mathbf{x}_N\}$
  - Dimensionality D
- Complexity
  - Storage: $O(N)$
  - Test runtime: $O(\log N)$
  - Training runtime: $O(DN^2 \log N)$
    - Most expensive part.
    - Critical step: selecting the optimal splitting point.
    - Need to check D dimensions, for each need to sort N data points.

$$O(DN \log N)$$

Recap: Decision Trees - Summary

- Properties
  - Simple learning procedure, fast evaluation.
  - Can be applied to metric, nominal, or mixed data.
  - Often yield interpretable results.

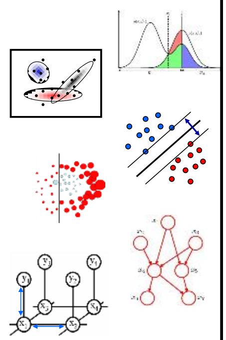
Recap: Decision Trees - Summary

- Limitations
  - Often produce noisy (bushy) or weak (stunted) classifiers.
  - Do not generalize too well.
  - Training data fragmentation:
    - As tree progresses, splits are selected based on less and less data.
  - Overtraining and undertraining:
    - Deep trees: fit the training data well, will not generalize well to new test data.
    - Shallow trees: not sufficiently refined.
  - Stability
    - Trees can be very sensitive to details of the training points.
    - If a single data point is only slightly shifted, a radically different tree may come out!

    ⇒ Result of discrete and greedy learning procedure.
  - Expensive learning step
    - Mostly due to costly selection of optimal split.

Course Outline

- Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation
  - Mixture Models and EM
- Discriminative Approaches
  - Linear Discriminant Functions
  - Statistical Learning Theory & SVMs
  - Ensemble Methods & Boosting
  - Decision Trees & Randomized Trees
- Generative Models
  - Bayesian Networks
  - Markov Random Fields
  - Exact Inference

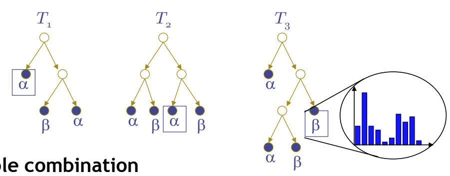


Recap: Randomized Decision Trees

- Decision trees: main effort on finding good split
  - Training runtime: $O(DN^2 \log N)$
  - This is what takes most effort in practice.
  - Especially cumbersome with many attributes (large D).
- Idea: randomize attribute selection
  - No longer look for globally optimal split.
  - Instead randomly use subset of K attributes on which to base the split.
  - Choose best splitting attribute e.g. by maximizing the information gain (= reducing entropy):

$$\triangle E = \sum_{k=1}^{K} \frac{|S_k|}{|S|} \sum_{j=1}^{N} p_j \log_2(p_j)$$

Recap: Ensemble Combination



- Ensemble combination
  - Tree leaves (l,η) store posterior probabilities of the target classes.

$$p_{l,\eta}(\mathcal{C}|\mathbf{x})$$

  - Combine the output of several trees by averaging their posteriors (Bayesian model combination)

$$p(\mathcal{C}|\mathbf{x}) = \frac{1}{L}\sum_{l=1}^{L} p_{l,\eta}(\mathcal{C}|\mathbf{x})$$

Recap: Random Forests (Breiman 2001)

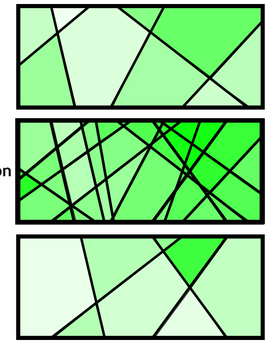
see Exercise 3.5

- General ensemble method
  - Idea: Create ensemble of many (50 - 1,000) trees.
- Empirically very good results
  - Often as good as SVMs (and sometimes better)!
  - Often as good as Boosting (and sometimes better)!
- Injecting randomness
  - Bootstrap sampling process
    - On average only 63% of training examples used for building the tree
    - Remaining 37% out-of-bag samples used for validation.
  - Random attribute selection
    - Randomly choose subset of K attributes to select from at each node.
    - Faster training procedure.
- Simple majority vote for tree combination

Recap: A Graphical Interpretation

Different trees induce different partitions on the data.

By combining them, we obtain a finer subdivision of the feature space…

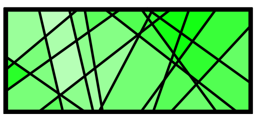


Slide credit: Vincent Lepetit

Recap: A Graphical Interpretation

Different trees induce different partitions on the data.

By combining them, we obtain a finer subdivision of the feature space…

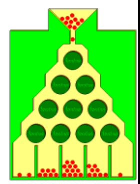


…which at the same time also better reflects the uncertainty due to the bootstrapped sampling.

Slide credit: Vincent Lepetit

Recap: Extremely Randomized Decision Trees

- Random queries at each node…
  - Tree gradually develops from a classifier to a flexible container structure.
  - Node queries define (randomly selected) structure.
  - Each leaf node stores posterior probabilities

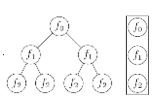


- Learning
  - Patches are "dropped down" the trees.
    - Only pairwise pixel comparisons at each node.
    - Directly update posterior distributions at leaves
  - ⇒ Very fast procedure, only few pixel-wise comparisons.
  - ⇒ No need to store the original patches!

Image source: Wikipedia

Recap: Ferns

- Ferns
  - Ferns are semi-naïve Bayes classifiers.
  - They assume independence between sets of features (between the ferns)…
  - …and enumerate all possible outcomes inside each set.



- Interpretation
  - Combine the tests $f_l,\ldots,f_{l+S}$ into a binary number.
  - Update the "fern leaf" corresponding to that number.

0
0 ⟶ Update leaf 100_2 = 4
1

Recap: Ferns (Semi-Naïve Bayes Classifiers)

- Ferns
  - A fern F is defined as a set of S binary features $\{f_l,\ldots,f_{l+S}\}$.
  - M: number of ferns, $N_f = S \cdot M$.
  - This represents a compromise:

$$p(f_1,\ldots,f_{N_f}|\mathcal{C}_k) \approx \prod_{j=1}^{M} p(F_j|\mathcal{C}_k)$$

$$= \underbrace{p(f_1,\ldots,f_S|\mathcal{C}_k)}_{\text{Full joint inside fern}} \cdot p(f_{S+1},\ldots,f_{2S}|\mathcal{C}_k) \cdot \ldots$$

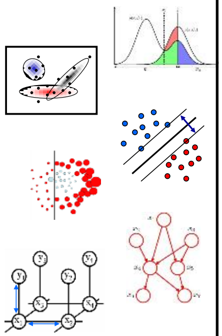
Naïve Bayes between ferns

⇒ Model with $M \cdot 2^S$ parameters ("Semi-Naïve").

⇒ Flexible solution that allows complexity/performance tuning.

Course Outline

- Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation
  - Mixture Models and EM
- Discriminative Approaches
  - Linear Discriminant Functions
  - Statistical Learning Theory & SVMs
  - Ensemble Methods & Boosting
  - Decision Trees & Randomized Trees
- Generative Models
  - Bayesian Networks
  - Markov Random Fields
  - Exact Inference

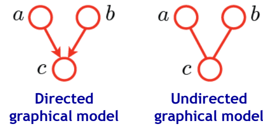


Machine Learning, Summer '15

B. Leibe

Recap: Graphical Models

- Two basic kinds of graphical models
  - Directed graphical models or Bayesian Networks
  - Undirected graphical models or Markov Random Fields
- Key components
  - Nodes
    - Random variables
  - Edges
    - Directed or undirected



  - The value of a random variable may be known or unknown.

unknown known

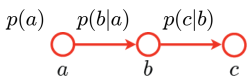
Machine Learning, Summer '15

Slide credit: Bernt Schiele

B. Leibe

Recap: Directed Graphical Models

- Chains of nodes:



  - Knowledge about a is expressed by the prior probability:
    $p(a)$
  - Dependencies are expressed through conditional probabilities:
    $p(b|a),\ p(c|b)$
  - Joint distribution of all three variables:

$$\begin{aligned} p(a,b,c) &= p(c|a,b)p(a,b) \\ &= p(c|b)p(b|a)p(a) \end{aligned}$$

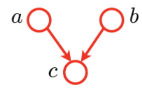
Machine Learning, Summer '15

Slide credit: Bernt Schiele, Stefan Roth

B. Leibe

Recap: Directed Graphical Models

- Convergent connections:



  - Here the value of c depends on both variables a and b.
  - This is modeled with the conditional probability:
    $p(c|a,b)$
  - Therefore, the joint probability of all three variables is given as:

$$\begin{aligned} p(a,b,c) &= p(c|a,b)p(a,b) \\ &= p(c|a,b)p(a)p(b) \end{aligned}$$

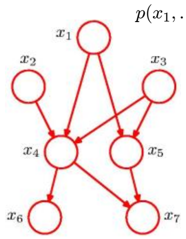
Machine Learning, Summer '15

Slide credit: Bernt Schiele, Stefan Roth

B. Leibe

Recap: Factorization of the Joint Probability

- Computing the joint probability



$$\begin{aligned} p(x_1,\ldots,x_7) = {} & p(x_1)p(x_2)p(x_3)p(x_4|x_1,x_2,x_3) \\ & p(x_5|x_1,x_3)p(x_6|x_4)p(x_7|x_4,x_5) \end{aligned}$$

General factorization

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k|\mathrm{pa}_k)$$

We can directly read off the factorization of the joint from the network structure!

Machine Learning, Summer '15

B. Leibe

Image source: C. Bishop, 2006

Recap: Factorized Representation

- Reduction of complexity
  - Joint probability of n binary variables requires us to represent values by brute force
    $\mathcal{O}(2^n)$ terms
  - The factorized form obtained from the graphical model only requires
    $\mathcal{O}(n \cdot 2^k)$ terms
    - k: maximum number of parents of a node.

⇒ It's the edges that are missing in the graph that are important! They encode the simplifying assumptions we make.

Machine Learning, Summer '15

Slide credit: Bernt Schiele, Stefan Roth

B. Leibe

Recap: Conditional Independence

see Exercise 4.2

- X is conditionally independent of Y given V
  - Definition: $X \perp\!\!\!\perp Y | V \quad \Leftrightarrow \quad p(X|Y,V) = p(X|V)$
  - Also: $X \perp\!\!\!\perp Y | V \quad \Leftrightarrow \quad p(X,Y|V) = p(X|V)\,p(Y|V)$
  - Special case: Marginal Independence

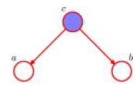
    $$X \perp\!\!\!\perp Y \quad \Leftrightarrow \quad X \perp\!\!\!\perp Y | \emptyset \quad \Leftrightarrow \quad p(X,Y) = p(X)\,p(Y)$$

  - Often, we are interested in conditional independence between sets of variables:

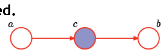
    $$\mathcal{X} \perp\!\!\!\perp \mathcal{Y} | \mathcal{V} \quad \Leftrightarrow \quad \{X \perp\!\!\!\perp Y | \mathcal{V},\ \forall X \in \mathcal{X} \text{ and } \forall Y \in \mathcal{Y}\}$$

Recap: Conditional Independence

- Three cases
  - Divergent ("Tail-to-Tail")
    - Conditional independence when c is observed.



  - Chain ("Head-to-Tail")
    - Conditional independence when c is observed.



  - Convergent ("Head-to-Head")
    - Conditional independence when neither c, nor any of its descendants are observed.

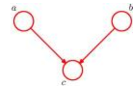
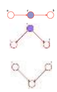


Image source: C. Bishop, 2006

Recap: D-Separation

see Exercise 4.3

- Definition
  - Let A, B, and C be non-intersecting subsets of nodes in a directed graph.
  - A path from A to B is blocked if it contains a node such that either
    - The arrows on the path meet either head-to-tail or tail-to-tail at the node, and the node is in the set C, or
    - The arrows meet head-to-head at the node, and neither the node, nor any of its descendants, are in the set C.



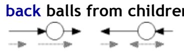
  - If all paths from A to B are blocked, A is said to be d-separated from B by C.
- If A is d-separated from B by C, the joint distribution over all variables in the graph satisfies $A \perp\!\!\!\perp B \mid C$.
  - Read: "A is conditionally independent of B given C."

Slide adapted from Chris Bishop

Recap: "Bayes Ball" Algorithm

see Exercise 4.5

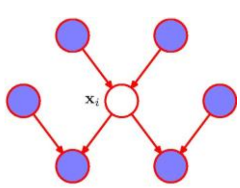
- Graph algorithm to compute d-separation
  - *Goal: Get a ball from X to Y without being blocked by $\mathcal{V}$.*
  - Depending on its direction and the previous node, the ball can
    - Pass through (from parent to all children, from child to all parents)
    - Bounce back (from any parent/child to all parents/children)
    - Be blocked
- Game rules
  - An unobserved node ($W \notin \mathcal{V}$) passes through balls from parents, but *also* bounces back balls from children.



  - An observed node ($W \in \mathcal{V}$) bounces back balls from parents, but blocks balls from children.

Slide adapted from Zoubin Gharahmani

Recap: The Markov Blanket

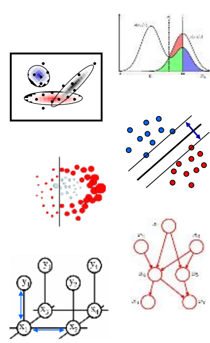


- Markov blanket of a node $\mathbf{x}_i$
  - Minimal set of nodes that isolates $\mathbf{x}_i$ from the rest of the graph.
  - This comprises the set of
    - Parents,
    - Children, and
    - Co-parents of $\mathbf{x}_i$. ← This is what we have to watch out for!

Image source: C. Bishop, 2006

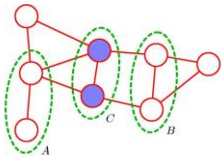
Course Outline

- Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation
  - Mixture Models and EM
- Discriminative Approaches
  - Linear Discriminant Functions
  - Statistical Learning Theory & SVMs
  - Ensemble Methods & Boosting
  - Decision Trees & Randomized Trees
- Generative Models
  - Bayesian Networks
  - Markov Random Fields
  - Exact Inference

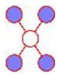


Recap: Undirected Graphical Models

- Undirected graphical models ("Markov Random Fields")
  - Given by undirected graph



- Conditional independence for undirected graphs
  - If every path from any node in set A to set B passes through at least one node in set C, then $A \perp\!\!\!\perp B \mid C$.
  - Simple Markov blanket:



Recap: Factorization in MRFs

- Joint distribution
  - Written as product of potential functions over maximal cliques in the graph:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_C \psi_C(\mathbf{x}_C)$$

  - The normalization constant Z is called the partition function.

$$Z = \sum_{\mathbf{x}} \prod_C \psi_C(\mathbf{x}_C)$$

- Remarks
  - BNs are automatically normalized. But for MRFs, we have to explicitly perform the normalization.
  - Presence of normalization constant is major limitation!
    - Evaluation of Z involves summing over $\mathcal{O}(K^M)$ terms for M nodes!

Factorization in MRFs

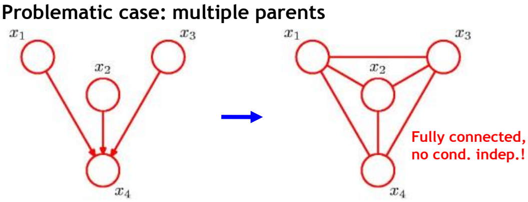
- Role of the potential functions
  - General interpretation
    - No restriction to potential functions that have a specific probabilistic interpretation as marginals or conditional distributions.
  - Convenient to express them as exponential functions ("Boltzmann distribution")

$$\psi_C(\mathbf{x}_C) = \exp\{-E(\mathbf{x}_C)\}$$

    - with an energy function E.
  - Why is this convenient?
    - Joint distribution is the product of potentials ⇒ sum of energies.
    - We can take the log and simply work with the sums…

Recap: Converting Directed to Undirected Graphs

- Problematic case: multiple parents



$$p(\mathbf{x}) = p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)$$

Need a clique of $x_1,\ldots,x_4$ to represent this factor!

  - Need to introduce additional links ("marry the parents").

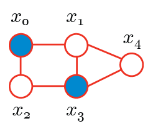
⇒ This process is called moralization. It results in the moral graph.

Recap: Conversion Algorithm

- General procedure to convert directed → undirected
  1. Add undirected links to marry the parents of each node.
  2. Drop the arrows on the original links ⇒ moral graph.
  3. Find maximal cliques for each node and initialize all clique potentials to 1.
  4. Take each conditional distribution factor of the original directed graph and multiply it into one clique potential.
- Restriction
  - Conditional independence properties are often lost!
  - Moralization results in additional connections and larger cliques.

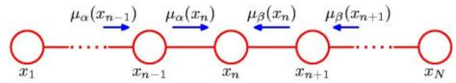
Recap: Computing Marginals

- How do we apply graphical models?
  - Given some observed variables, we want to compute distributions of the unobserved variables.
  - In particular, we want to compute marginal distributions, for example $p(x_4)$.



- How can we compute marginals?
  - Classical technique: sum-product algorithm by Judea Pearl.
  - In the context of (loopy) undirected models, this is also called (loopy) belief propagation [Weiss, 1997].
  - Basic idea: message-passing.

Recap: Message Passing on a Chain



- Idea
  - Pass messages from the two ends towards the query node x_n.
- Define the messages recursively:

$$\mu_\alpha(x_n) = \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n)\mu_\alpha(x_{n-1})$$

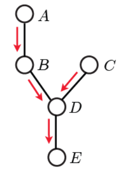
$$\mu_\beta(x_n) = \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1})\mu_\beta(x_{n+1})$$

- Compute the normalization constant Z at any node x_m.

$$Z = \sum_{x_n} \mu_\alpha(x_n)\mu_\beta(x_n)$$

Recap: Message Passing on Trees

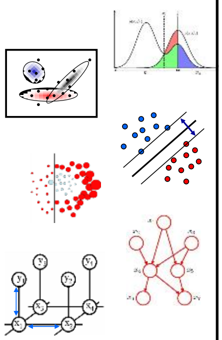
- General procedure for all tree graphs.
  - Root the tree at the variable that we want to compute the marginal of.
  - Start computing messages at the leaves.
  - Compute the messages for all nodes for which all incoming messages have already been computed.
  - Repeat until we reach the root.



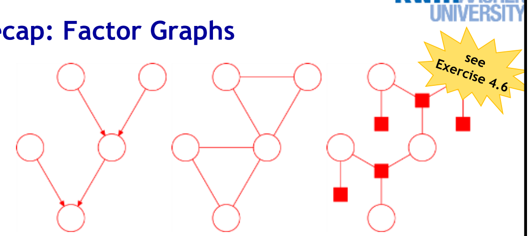
- If we want to compute the marginals for all possible nodes (roots), we can reuse some of the messages.
  - Computational expense linear in the number of nodes.
- We already motivated message passing for inference.
  - How can we formalize this into a general algorithm?

Course Outline

- Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation
  - Mixture Models and EM
- Discriminative Approaches
  - Linear Discriminant Functions
  - Statistical Learning Theory & SVMs
  - Ensemble Methods & Boosting
  - Decision Trees & Randomized Trees
- Generative Models
  - Bayesian Networks
  - Markov Random Fields
  - Exact Inference



Recap: Factor Graphs



- Joint probability
  - Can be expressed as product of factors: $p(\mathbf{x}) = \frac{1}{Z}\prod_s f_s(\mathbf{x}_s)$
  - Factor graphs make this explicit through separate factor nodes.
- Converting a directed polytree
  - Conversion to undirected tree creates loops due to moralization!
  - Conversion to a factor graph again results in a tree!

Recap: Sum-Product Algorithm

- Objectives
  - Efficient, exact inference algorithm for finding marginals.
- Procedure:
  - Pick an arbitrary node as root.
  - Compute and propagate messages from the leaf nodes to the root, storing received messages at every node.
  - Compute and propagate messages from the root to the leaf nodes, storing received messages at every node.
  - Compute the product of received messages at each node for which the marginal is required, and normalize if necessary.

$$p(x) \propto \prod_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x)$$

- Computational effort
  - Total number of messages = 2 · number of graph edges.

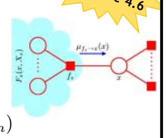
Recap: Sum-Product Algorithm

see Exercise 4.6

- Two kinds of messages
  - Message from factor node to variable nodes:
    - Sum of factor contributions

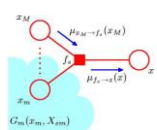
$$\mu_{f_s \to x}(x) \equiv \sum_{X_s} F_s(x, X_s)$$

$$= \sum_{X_s} f_s(\mathbf{x}_s) \prod_{m \in \mathrm{ne}(f_s)\setminus x} \mu_{x_m \to f_s}(x_m)$$



  - Message from variable node to factor node:
    - Product of incoming messages

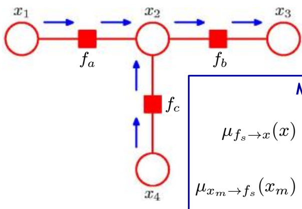
$$\mu_{x_m \to f_s}(x_m) \equiv \prod_{l \in \mathrm{ne}(x_m)\setminus f_s} \mu_{f_l \to x_m}(x_m)$$



⇒ Simple propagation scheme.

Recap: Sum-Product from Leaves to Root

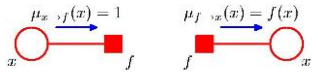
see Exercise 4.6



Message definitions:

$$\mu_{f_s \to x}(x) \equiv \sum_{X_s} f_s(\mathbf{x}_s) \prod_{m \in \mathrm{ne}(f_s)\setminus x} \mu_{x_m \to f_s}(x_m)$$

$$\mu_{x_m \to f_s}(x_m) \equiv \prod_{l \in \mathrm{ne}(x_m)\setminus f_s} \mu_{f_l \to x_m}(x_m)$$



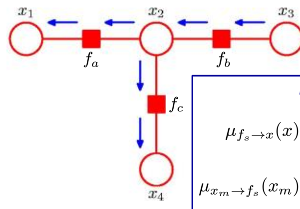
B. Leibe

109

Image source: C. Bishop, 2006

Machine Learning, Summer '15

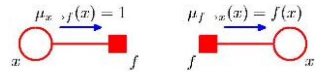
Recap: Sum-Product from Root to Leaves



Message definitions:

$$\mu_{f_s \to x}(x) \equiv \sum_{X_s} f_s(\mathbf{x}_s) \prod_{m \in \mathrm{ne}(f_s)\setminus x} \mu_{x_m \to f_s}(x_m)$$

$$\mu_{x_m \to f_s}(x_m) \equiv \prod_{l \in \mathrm{ne}(x_m)\setminus f_s} \mu_{f_l \to x_m}(x_m)$$



B. Leibe

110

Image source: C. Bishop, 2006

Machine Learning, Summer '15

Recap: Max-Sum Algorithm

- Objective: an efficient algorithm for finding
  - Value $\mathbf{x}^{\max}$ that maximises $p(\mathbf{x})$;
  - Value of $p(\mathbf{x}^{\max})$.
  - ⇒ Application of dynamic programming in graphical models.
- Key ideas
  - We are interested in the maximum value of the joint distribution

$$p(\mathbf{x}^{\max}) = \max_{\mathbf{x}} p(\mathbf{x})$$

  - ⇒ Maximize the product $p(\mathbf{x})$.
  - For numerical reasons, use the logarithm.

$$\ln\left(\max_{\mathbf{x}} p(\mathbf{x})\right) = \max_{\mathbf{x}} \ln p(\mathbf{x}).$$

  - ⇒ Maximize the sum (of log-probabilities).

Slide adapted from Chris Bishop

B. Leibe

111

Machine Learning, Summer '15

Recap: Max-Sum Algorithm

- Initialization (leaf nodes)

$$\mu_{x \to f}(x) = 0 \qquad \mu_{f \to x}(x) = \ln f(x)$$

- Recursion
  - Messages

$$\mu_{f \to x}(x) = \max_{x_1,\ldots,x_M}\left[\ln f(x, x_1, \ldots, x_M) + \sum_{m \in \mathrm{ne}(f_s)\setminus x} \mu_{x_m \to f}(x_m)\right]$$

$$\mu_{x \to f}(x) = \sum_{l \in \mathrm{ne}(x)\setminus f} \mu_{f_l \to x}(x)$$

  - For each node, keep a record of which values of the variables gave rise to the maximum state:

$$\phi(x) = \arg\max_{x_1,\ldots,x_M}\left[\ln f(x, x_1, \ldots, x_M) + \sum_{m \in \mathrm{ne}(f_s)\setminus x} \mu_{x_m \to f}(x_m)\right]$$

Slide adapted from Chris Bishop

B. Leibe

112

Machine Learning, Summer '15

Recap: Max-Sum Algorithm

- Termination (root node)
  - Score of maximal configuration

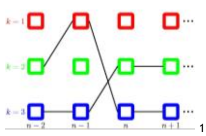
$$p^{\max} = \max_x \left[\sum_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x)\right]$$

  - Value of root node variable giving rise to that maximum

$$x^{\max} = \arg\max_x \left[\sum_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x)\right]$$

  - Back-track to get the remaining variable values

$$x_{n-1}^{\max} = \phi(x_n^{\max})$$



Slide adapted from Chris Bishop

B. Leibe

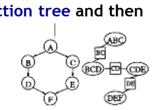
113

Machine Learning, Summer '15

Recap: Junction Tree Algorithm

see Exercise 5.1

- Motivation
  - Exact inference on general graphs.
  - Works by turning the initial graph into a junction tree and then running a sum-product-like algorithm.
  - Intractable on graphs with large cliques.



- Main steps
  1. If starting from directed graph, first convert it to an undirected graph by moralization.
  2. Introduce additional links by triangulation in order to reduce the size of cycles.
  3. Find cliques of the moralized, triangulated graph.
  4. Construct a new graph from the maximal cliques.
  5. Remove minimal links to break cycles and get a junction tree.

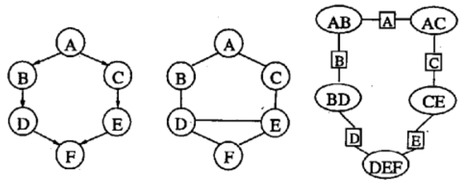
  ⇒ Apply regular message passing to perform inference.

B. Leibe

114

Machine Learning, Summer '15

Recap: Junction Tree Example

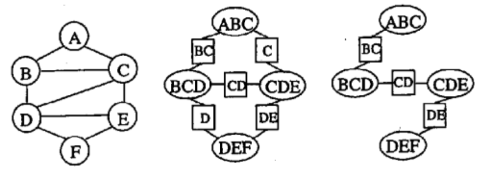


- Without triangulation step
  - The final graph will contain cycles that we cannot break without losing the running intersection property!

Image source: J. Pearl, 1988

RWTH AACHEN UNIVERSITY

Recap: Junction Tree Example



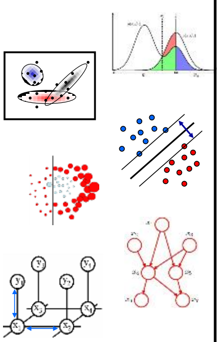
- When applying the triangulation
  - Only small cycles remain that are easy to break.
  - Running intersection property is maintained.

Image source: J. Pearl, 1988

RWTH AACHEN UNIVERSITY

Course Outline

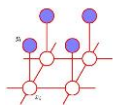
- Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation
  - Mixture Models and EM
- Discriminative Approaches
  - Linear Discriminant Functions
  - Statistical Learning Theory & SVMs
  - Ensemble Methods & Boosting
  - Decision Trees & Randomized Trees
- Generative Models
  - Bayesian Networks
  - Markov Random Fields & Applications
  - Exact Inference



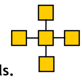
RWTH AACHEN UNIVERSITY

Recap: MRF Structure for Images

- Basic structure



- Two components
  - Observation model
    - How likely is it that node x_i has label L_i given observation y_i?
    - This relationship is usually learned from training data.
  - Neighborhood relations
    - Simplest case: 4-neighborhood
    - Serve as smoothing terms.
    - ⇒ Discourage neighboring pixels to have different labels.
    - This can either be learned or be set to fixed "penalties".



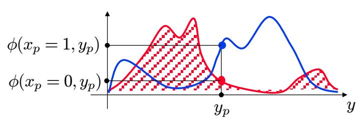
RWTH AACHEN UNIVERSITY

Recap: How to Set the Potentials?

- Unary potentials
  - E.g. color model, modeled with a Mixture of Gaussians

$$\phi(x_i, y_i; \theta_\phi) = \log \sum_k \theta_\phi(x_i, k) p(k|x_i) \mathcal{N}(y_i; \bar{y}_k, \Sigma_k)$$

⇒ Learn color distributions for each label



RWTH AACHEN UNIVERSITY

Recap: How to Set the Potentials?

- Pairwise potentials
  - Potts Model

$$\psi(x_i, x_j; \theta_\psi) = \theta_\psi \delta(x_i \neq x_j)$$

    - Simplest discontinuity preserving model.
    - Discontinuities between any pair of labels are penalized equally.
    - Useful when labels are unordered or number of labels is small.
  - Extension: "contrast sensitive Potts model"

$$\psi(x_i, x_j, g_{ij}(y); \theta_\psi) = \theta_\psi g_{ij}(y) \delta(x_i \neq x_j)$$

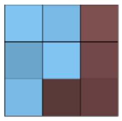
where

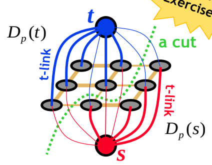
$$g_{ij}(y) = e^{-\beta \|y_i - y_j\|^2} \qquad \beta = 2 / avg\left(\|y_i - y_j\|^2\right)$$

    - Discourages label changes except in places where there is also a large change in the observations.

Recap: Graph Cuts for Binary Problems

see Exercise 5.3





"expected" intensities of object and background I^s and I^t can be re-estimated

$$D_p(s) \propto \exp\left(-\| I_p - I^s \|^2 / 2\sigma^2\right)$$
$$D_p(t) \propto \exp\left(-\| I_p - I^t \|^2 / 2\sigma^2\right)$$

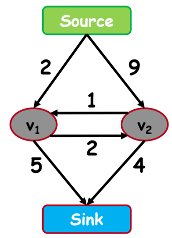
EM-style optimization

Slide credit: Yuri Boykov [Boykov & Jolly, ICCV'01]

Recap: s-t-Mincut Equivalent to Maxflow

see Exercise 5.2

Flow = 0



Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli

Recap: When Can s-t Graph Cuts Be Applied?

$$E(L) = \underbrace{\sum_p E_p(L_p)}_{\text{Regional term (t-links)}} + \underbrace{\sum_{pq \in N} E(L_p, L_q)}_{\text{Boundary term (n-links)}} \qquad L_p \in \{s,t\}$$

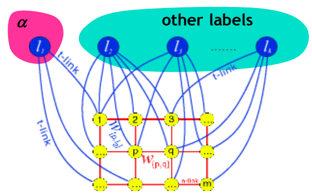
- s-t graph cuts can only globally minimize binary energies that are submodular. [Boros & Hummer, 2002, Kolmogorov & Zabih, 2004]

E(L) can be minimized by *s-t* graph cuts $\Leftrightarrow$ $E(s,s)+E(t,t) \leq E(s,t)+E(t,s)$ Submodularity ("convexity")

- Submodularity is the discrete equivalent to convexity.
  - Implies that every local energy minimum is a global minimum.
  - $\Rightarrow$ Solution will be globally optimal.

Recap: α-Expansion Move

- Basic idea:
  - Break multi-way cut computation into a sequence of binary s-t cuts.



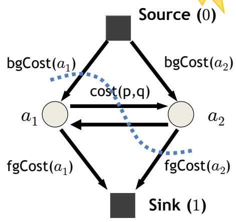
  - No longer globally optimal result, but guaranteed approximation quality and typically converges in few iterations.

Slide credit: Yuri Boykov

Recap: Converting an MRF to an s-t Graph

see Exercise 5.4

```
Graph *g;
For all pixels p
    /* Add a node to the graph */
    nodeID(p) = g->add_node();
    /* Set cost of terminal edges */
    set_weights(nodeID(p), fgCost(p), bgCost(p));
end
for all adjacent pixels p,q
    add_weights(nodeID(p), nodeID(q), cost);
end
g->compute_maxflow();
label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```



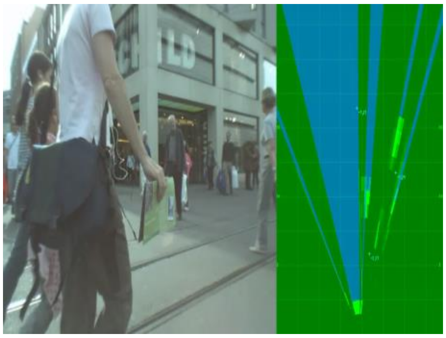
a_1 = bg a_2 = fg

Slide credit: Pushmeet Kohli

Any Questions?

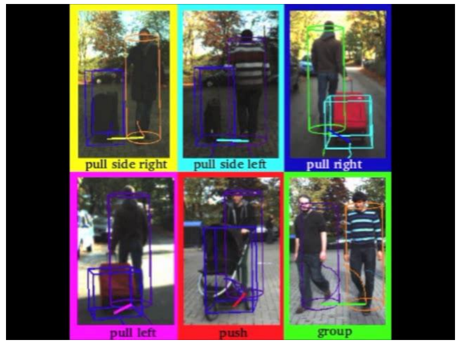
So what can you do with all of this?

Mobile Object Detection & Tracking



[Ess, Leibe, Schindler, Van Gool, CVPR'08]

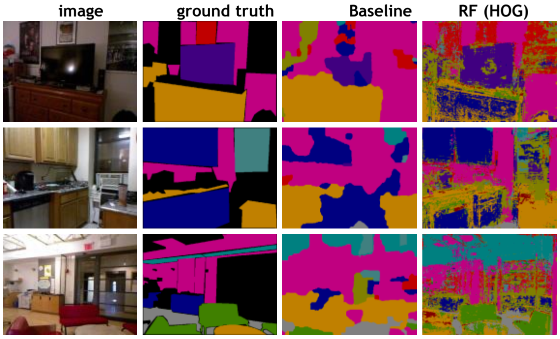
Learning Person-Object Interactions



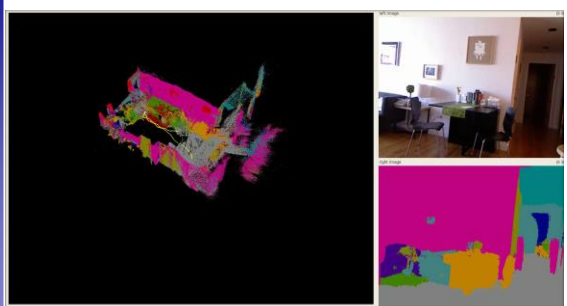
B. Leibe [T. Baumgartner, D. Mitzel, B. Leibe, CVPR'13]

Semantic Segmentation

image | ground truth | Baseline | RF (HOG)



3D Labeling Results - Living Room



play video

[Hermans, Floros, Leibe, submission to ICCV'13]

Semantic Scene Segmentation

B. Leibe [G. Floros, B. Leibe, CVPR'12]

Any More Questions?

Good luck for the exam!