Machine Learning - Lecture 19

Repetition

21.07.2010

Bastian Leibe
RWTH Aachen
http://www.mmp.rwth-aachen.de
leibe@umic.rwth-aachen.de
Announcements

- Did everybody register for the exam dates on the Doodle poll?
  - If not, please contact me.
  - Also if there are problems with the proposed dates...
Lecture Evaluation: Positive Comments

• “Good structure, good explanation, excellent lecturer. Very good course!”

• “Despite the highly complex topic the lecture is very good to follow. Very good course.”
Lecture Evaluation: Detailed Critique

• “Focus too much on graphical models for a ML lecture. Deeper treatment of statistical learning theory & SVMs or perhaps neural networks & perceptron algorithm would be more interesting. Everything else is good.”
Lecture Evaluation: Exercises

• “Exercises take a lot of time.”

• “Übungen an sich gut, aber zu umfangreich”.

• “Workload for first homeworks too high”.

• “The lecture slides sometimes don’t detail formulas enough for successful completion of the exercises (e.g., Sum-Product algorithm, CART framework)”.

• “Difficult to transfer algorithms from class to code: sometimes variables not clear (e.g., b in SVM). Pseudo-code could help.”
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.

• Oral exam procedure
  ➢ Oral exam, form depends on B.Sc./M.Sc./Diplom specifics
  ➢ Procedure: 3 (or 4) questions, will have to answer 2 (3) of them
  ➢ For 4 credits, supplementary material also required.
Course Outline

• Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation
  - Mixture Models and EM

• Discriminative Approaches
  - Lin. Discriminants, SVMs, Boosting

• Generative Models
  - Bayesian Networks
  - Markov Random Fields
  - Exact & Approximate Inference

• Unifying Perspective
  - HMMs, EM, CRFs, ...
Recap: Bayes Decision Theory

\[ p(x \mid a) \]

\[ p(x \mid b) \]

\[ p(x \mid a) \, p(a) \]

\[ p(x \mid b) \, p(b) \]

\[ \text{Decision boundary} \]

\[ p(a \mid x) \]

\[ p(b \mid x) \]

\[ \text{Likelihood} \times \text{Prior} \]

\[ \text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Normalization Factor}} \]
Recap: Bayes Decision Theory

- Optimal decision rule
  - Decide for $C_1$ if
    $$p(C_1 | x) > p(C_2 | x)$$
  - This is equivalent to
    $$p(x | C_1) p(C_1) > p(x | C_2) p(C_2)$$
  - Which is again equivalent to (Likelihood-Ratio test)
    $$\frac{p(x | C_1)}{p(x | C_2)} > \frac{p(C_2)}{p(C_1)}$$
    Decision threshold $\theta$

Slide credit: Bernt Schiele
Recap: Bayes Decision Theory

- Decision regions: $R_1$, $R_2$, $R_3$, ...
Recap: Classifying with Loss Functions

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

$$L_{kj} = \text{loss for decision } C_j \text{ if truth is } C_k.$$  

- Example: cancer diagnosis

$$L_{\text{cancer diagnosis}} = \begin{pmatrix}
\text{Truth} & \text{cancer} & \text{normal} \\
\text{cancer} & 0 & 1000 \\
\text{normal} & 1 & 0
\end{pmatrix}$$
Recap: Minimizing the Expected Loss

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

- Solution: Minimize the expected loss

  \[ \mathbb{E}[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{k,j} p(x, C_k) \, dx \]

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

  \[ \mathbb{E}[L] = \sum_k L_{k,j} p(C_k|x) \]

  which is easy to do once we know the posterior class probabilities \( p(C_k|x) \).
Recap: The Reject Option

- Classification errors arise from regions where the largest posterior probability $p(C_k | x)$ 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.

B. Leibe

Image source: C.M. Bishop, 2006
Course Outline

- **Fundamentals**
  - Bayes Decision Theory
  - Probability Density Estimation
  - Mixture Models and EM

- **Discriminative Approaches**
  - Lin. Discriminants, SVMs, Boosting

- **Generative Models**
  - Bayesian Networks
  - Markov Random Fields
  - Exact & Approximate Inference

- **Unifying Perspective**
  - HMMs, EM, CRFs, ...
Recap: Gaussian (or Normal) Distribution

- **One-dimensional case**
  - Mean $\mu$
  - Variance $\sigma^2$

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

- **Multi-dimensional case**
  - Mean $\mu$
  - Covariance $\Sigma$

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

Image source: C.M. Bishop, 2006
Recap: Maximum Likelihood Approach

- 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 \( \theta \) (Learning)
  - Maximize the likelihood (= minimize the negative log-likelihood)
    ⇒ Take the derivative and set it to zero.

\[
\frac{\partial}{\partial \theta} E(\theta) = - \sum_{n=1}^{N} \frac{\partial}{\partial \theta} p(x_n|\theta) \frac{1}{p(x_n|\theta)} = 0
\]
Recap: Bayesian Learning Approach

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

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

Assumption: given $\theta$, this doesn’t depend on $X$ anymore

\[
p(x, \theta|X) = p(x|\theta, X)p(\theta|X)
\]

\[
p(x|X) = \int_p(x|\theta)p(\theta|X) d\theta
\]

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

Slide adapted from Bernt Schiele
Recap: Bayesian Learning Approach

- **Discussion**

  
  Likelihood of the parametric form \( \theta \) given the data set \( X \).

  Estimate for \( x \) based on parametric form \( \theta \)

  Prior for the parameters \( \theta \)

  Normalization: integrate over all possible values of \( \theta \)

  - The more uncertain we are about \( \theta \), the more we average over all possible parameter values.

  \[
p(x|X) = \int \frac{p(x|\theta)L(\theta)p(\theta)}{\int L(\theta)p(\theta)d\theta} d\theta
  \]
Recap: Histograms

- Basic idea:
  - Partition the data space into distinct bins with widths $\Delta_i$ 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 $D$!

Image source: C.M. Bishop, 2006
Recap: Kernel Density Estimation

- Approximation formula:
  \[ p(x) \approx \frac{K}{NV} \]

Kernel Methods

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

K-Nearest Neighbor

- 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**
  - Lin. Discriminants, SVMs, Boosting

- **Generative Models**
  - Bayesian Networks
  - Markov Random Fields
  - Exact & Approximate Inference

- **Unifying Perspective**
  - HMMs, EM, CRFs, ...
Recap: Mixture of Gaussians (MoG)

- “Generative model”

\[
p(j) = \pi_j
\]

“Weight” of mixture component

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...

\[
\begin{align*}
\mu_1 &= \frac{\sum_{n=1}^{N} h(j = 1|x_n)x_n}{\sum_{i=1}^{N} h(j = 1|x_n)} \\
\mu_2 &= \frac{\sum_{n=1}^{N} h(j = 2|x_n)x_n}{\sum_{i=1}^{N} h(j = 2|x_n)}
\end{align*}
\]

ML for Gaussian #1

\[
h(j = 1|x_n) = \begin{cases} 1 & \text{if } j = 1 \\ 0 & \text{otherwise} \end{cases}
\]

ML for Gaussian #2

\[
h(j = 2|x_n) = \begin{cases} 1 & \text{if } j = 2 \\ 0 & \text{otherwise} \end{cases}
\]
Recap: MoG - Iterative Strategy

- Assuming we knew the mixture components...

\[ p(j = 1|x) \quad \text{assumed known} \]

\[ p(j = 2|x) \]

- 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.

Slide credit: Bernt Schiele
Recap: EM Algorithm

- **Expectation-Maximization (EM) Algorithm**
  - **E-Step**: softly assign samples to mixture components
    \[
    \gamma_j(x_n) \leftarrow \frac{\pi_j \mathcal{N}(x_n | \mu_j, \Sigma_j)}{\sum_{k=1}^{N} \pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k)} \quad \forall j = 1, \ldots, K, \ n = 1, \ldots, N
    \]
  - **M-Step**: re-estimate the parameters (separately for each mixture component) based on the soft assignments
    \[
    \hat{\pi}_j \leftarrow \frac{\hat{N}_j}{N} = \text{soft number of samples labeled } j
    \]
    \[
    \hat{\mu}_j^{\text{new}} \leftarrow \frac{1}{\hat{N}_j} \sum_{n=1}^{N} \gamma_j(x_n)x_n
    \]
    \[
    \hat{\Sigma}_j^{\text{new}} \leftarrow \frac{1}{\hat{N}_j} \sum_{n=1}^{N} \gamma_j(x_n)(x_n - \hat{\mu}_j^{\text{new}})(x_n - \hat{\mu}_j^{\text{new}})^T
    \]

See Exercise 2.4

Slide adapted from Bernt Schiele
Course Outline

• Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation
  - Mixture Models and EM

• Discriminative Approaches
  - Linear Discriminant Functions
  - Statistical Learning Theory & SVMs
  - Boosting, Decision Trees

• Generative Models
  - Bayesian Networks
  - Markov Random Fields

• Unifying Perspective
  - HMMs, EM, CRFs, ...
Recap: Linear Discriminant Functions

- **Basic idea**
  - Directly encode decision boundary
  - Minimize misclassification probability directly.

- **Linear discriminant functions**
  \[ y(x) = w^T x + w_0 \]
  - **weight vector**
  - **“bias”** (= threshold)
  - \( w, w_0 \) 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

- Simplest approach
  - Directly try to minimize the sum-of-squares error
    \[ E(w) = \sum_{n=1}^{N} (y(x_n; w) - t_n)^2 \]
    \[ E_D(\tilde{W}) = \frac{1}{2} \text{Tr} \left\{ (\tilde{X}\tilde{W} - T)^T(\tilde{X}\tilde{W} - T) \right\} \]
  - Setting the derivative to zero yields
    \[ \tilde{W} = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^TT = \tilde{X}^\dagger T \]
  - We then obtain the discriminant function as
    \[ y(x) = \tilde{W}^T\tilde{x} = T^T(\tilde{X}^\dagger)^T\tilde{x} \]
  - \( \Rightarrow \) 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”.

B. Leibe

Image source: C.M. Bishop, 2006
Recap: Generalized Linear Models

- Generalized linear model
  \[ y(x) = g(w^T x + w_0) \]
  - \( g(\cdot) \) is called an activation function and may be nonlinear.
  - The decision surfaces correspond to\[ y(x) = \text{const.} \iff w^T x + w_0 = \text{const.} \]
  - If \( g \) is monotonous (which is typically the case), the resulting decision boundaries are still linear functions of \( x \).

- 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 \( y(x) \) as posterior probabilities.

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

B. Leibe
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 $\phi_j$, 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.
  
    $\Rightarrow$ Minimization with Gradient Descent
Course Outline

• Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation

• Discriminative Approaches
  - Linear Discriminant Functions
  - Statistical Learning Theory
  - Support Vector Machines
  - Boosting, Decision Trees

• Generative Models
  - Bayesian Networks
  - Markov Random Fields

• Unifying Perspective
  - HMMs, EM, CRFs, …
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
  
  \[ R_{emp}(\alpha) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i; \alpha)) \]

- **Actual risk (= Expected risk)**
  - Expectation of the error on all data.
  
  \[ R(\alpha) = \int L(y_i, f(x; \alpha))dP_{X,Y}(x, y) \]

- \( P_{X,Y}(x, y) \) is the probability distribution of \((x,y)\).
  - It is fixed, but typically unknown.
  - \( \Rightarrow \) 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
    \[ R(\alpha) \leq R_{emp}(\alpha) + \epsilon(N, p^*, h) \]
  - where
    \( N \): number of training examples
    \( p^* \): probability that the bound is correct
    \( h \): capacity of the learning machine (“VC-dimension”)
Recap: VC Dimension

- Vapnik-Chervonenkis dimension
  - Measure for the capacity of a learning machine.

- Formal definition:
  - If a given set of $\ell$ points can be labeled in all possible $2^\ell$ 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
    \[
    R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h(\log(2N/h) + 1) - \log(\eta/4)}{N}}
    \]
    \[\text{“VC confidence”}\]
  - This bound is independent of $P_{X,Y}(x, y)$!
  - If we know $h$ (the VC dimension), we can easily compute the risk bound
    \[
    R(\alpha) \leq R_{emp}(\alpha) + \epsilon(N, p^*, h)
    \]
Recap: Structural Risk Minimization

- How can we implement Structural Risk Minimization?
  \[ R(\alpha) \leq R_{emp}(\alpha) + \epsilon(N, p^*, h) \]

- Classic approach
  1. Keep \( \epsilon(N, p^*, h) \) constant and minimize \( R_{emp}(\alpha) \).
  2. \( \epsilon(N, p^*, h) \) can be kept constant by controlling the model parameters.

- Support Vector Machines (SVMs)
  1. Keep \( R_{emp}(\alpha) \) constant and minimize \( \epsilon(N, p^*, h) \).
  2. In fact: \( R_{emp}(\alpha) = 0 \) for separable data.
  3. 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

- **Discriminative Approaches**
  - Linear Discriminant Functions
  - Statistical Learning Theory
  - Support Vector Machines
  - Boosting, Decision Trees

- **Generative Models**
  - Bayesian Networks
  - Markov Random Fields

- **Unifying Perspective**
  - HMMs, EM, CRFs, ...
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
    \[ w^T x + b = 0 \]

- **Formulation as a convex optimization problem**
  - Find the hyperplane satisfying
    \[
    \arg\min_{w,b} \frac{1}{2} \| w \|^2
    \]
    under the constraints
    \[
    t_n (w^T x_n + b) \geq 1 \quad \forall n
    \]
    based on training data points \( x_n \) and target values \( t_n \in \{-1, 1\} \).

B. Leibe
Recap: SVM - Primal Formulation

- Lagrangian primal form

\[
L_p = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n \left\{ t_n (w^T x_n + b) - 1 \right\}
\]

\[
= \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n \left\{ t_n y(x_n) - 1 \right\}
\]

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

\[
a_n \geq 0
\]

\[
t_n y(x_n) - 1 \geq 0
\]

\[
a_n \left\{ t_n y(x_n) - 1 \right\} = 0
\]

\[
\begin{align*}
\text{KKT:} \\
\lambda \geq 0 \\
f(x) \geq 0 \\
\lambda f(x) = 0
\end{align*}
\]
Recap: SVM - Solution

- Solution for the hyperplane
  - Computed as a linear combination of the training examples
    \[ w = \sum_{n=1}^{N} a_n t_n x_n \]
  - 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:
    \[
    b = \frac{1}{NS} \sum_{n \in S} \left( t_n - \sum_{m \in S} a_m t_m x_m^T x_n \right)
    \]
Recap: SVM - Support Vectors

- The training points for which $\alpha_n > 0$ 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

- **Maximize**

\[ L_d(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (x_m^T x_n) \]

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 \( O(D^3) \).
  - \( L_d \) scales with \( O(N^3) \) - in practice between \( O(N) \) and \( O(N^2) \).

See Exercise 2.5
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.
  - $\xi_n = |t_n - y(x_n)|$ for all other points.

- We do not have to set the slack variables ourselves!
  $\Rightarrow$ They are jointly optimized together with $w$. 

B. Leibe
Recap: SVM - New Dual Formulation

- New SVM Dual: Maximize

\[ L_d(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (x_m^T x_n) \]

under the conditions

\[ 0 \leq a_n \leq C \]
\[ \sum_{n=1}^{N} a_n t_n = 0 \]

- This is again a quadratic programming problem

⇒ Solve as before...

This is all that changed!

Slide adapted from Bernt Schiele

B. Leibe
Recap: Nonlinear SVMs

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

$$\Phi: x \rightarrow \phi(x)$$
Recap: The Kernel Trick

- Important observation
  - \( \phi(x) \) only appears in the form of dot products \( \phi(x)^T \phi(y) \):
    \[
    y(x) = w^T \phi(x) + b
    \]
    \[
    = \sum_{n=1}^{N} a_n t_n \phi(x_n)^T \phi(x) + b
    \]
  - Define a so-called kernel function \( k(x,y) = \phi(x)^T \phi(y) \).
  - Now, in place of the dot product, use the kernel instead:
    \[
    y(x) = \sum_{n=1}^{N} a_n t_n k(x_n, x) + b
    \]
  - The kernel function *implicitly* maps the data to the higher-dimensional space (without having to compute \( \phi(x) \) explicitly)!
Recap: Kernels Fulfilling Mercer’s Condition

- **Polynomial kernel**
  \[ k(x, y) = (x^T y + 1)^p \]

- **Radial Basis Function kernel**
  \[ k(x, y) = \exp \left\{ -\frac{(x - y)^2}{2\sigma^2} \right\} \]
  e.g. Gaussian

- **Hyperbolic tangent kernel**
  \[ k(x, y) = \tanh(\kappa x^T y + \delta) \]
  e.g. Sigmoid

- And many, many more, including kernels on graphs, strings, and symbolic data...
Recap: Nonlinear SVM - Dual Formulation

- SVM Dual: Maximize
  \[ L_d(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(x_m, x_n) \]
  under the conditions
  \[ 0 \leq a_n \leq C \]
  \[ \sum_{n=1}^{N} a_n t_n = 0 \]

- Classify new data points using
  \[ y(x) = \sum_{n=1}^{N} a_n t_n k(x_n, x) + b \]
Course Outline

• Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation

• Discriminative Approaches
  - Linear Discriminant Functions
  - Statistical Learning Theory
  - Support Vector Machines
  - Boosting, Decision Trees

• Generative Models
  - Bayesian Networks
  - Markov Random Fields

• Unifying Perspective
  - HMMs, EM, CRFs, ...
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?
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.
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.
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
    \[
    p(X) = \sum_{h=1}^{H} p(X|h)p(h)
    \]

- Average error of committee
  - 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: Boosting (Schapire 1989)

- Algorithm: (3-component classifier)

1. Sample $N_1 < N$ training examples (without replacement) from training set $\mathcal{D}$ to get set $\mathcal{D}_1$.
   - Train weak classifier $C_1$ on $\mathcal{D}_1$.

2. Sample $N_2 < N$ training examples (without replacement), half of which were misclassified by $C_1$ to get set $\mathcal{D}_2$.
   - Train weak classifier $C_2$ on $\mathcal{D}_2$.

3. Choose all data in $\mathcal{D}$ on which $C_1$ and $C_2$ disagree to get set $\mathcal{D}_3$.
   - Train weak classifier $C_3$ on $\mathcal{D}_3$.

4. Get the final classifier output by majority voting of $C_1$, $C_2$, and $C_3$.
   (Recursively apply the procedure on $C_1$ to $C_3$)

B. Leibe

Image source: Duda, Hart, Stork, 2001
Recap: AdaBoost - “Adaptive Boosting”

- **Main idea**
  - 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**
  - $h_m(x)$: “weak” or base classifier
    - Condition: <50% training error over any distribution
  - $H(x)$: “strong” or final classifier

- **AdaBoost:**
  - Construct a strong classifier as a thresholded linear combination of the weighted weak classifiers:
    $$H(x) = \text{sign} \left( \sum_{m=1}^{M} \alpha_m h_m(x) \right)$$
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

Weak Classifier 1

Weights Increased

Weak Classifier 2

Figure adapted from Freund & Schapire
Recap: AdaBoost - Intuition

Final classifier is combination of the weak classifiers

Slide credit: Kristen Grauman

Figure adapted from Freund & Schapire
Recap: AdaBoost - Algorithm

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 $h_m(x)$ using the current weighting coefficients $W^{(m)}$ by minimizing the weighted error function

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

   where $I(A) = \begin{cases} 1, & \text{if } A \text{ is true} \\ 0, & \text{else} \end{cases}$

   b) Estimate the weighted error of this classifier on $X$:

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

   c) Calculate a weighting coefficient for $h_m(x)$:

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

   d) Update the weighting coefficients:

   $$w_n^{(m+1)} = w_n^{(m)} \exp \{\alpha_m I(h_m(x_n) \neq t_n)\}$$
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!

B. Leibe

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

- Discriminative Approaches
  - Linear Discriminant Functions
  - Statistical Learning Theory
  - Support Vector Machines
  - Boosting, Decision Trees

- Generative Models
  - Bayesian Networks
  - Markov Random Fields

- Unifying Perspective
  - HMMs, EM, CRFs, ...
Recap: Decision Trees

- Example:
  - “Classify Saturday mornings according to whether they’re suitable for playing tennis.”

B. Leibe

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

- **Goal**
  - Select the query (=split) that decreases impurity the most
  \[
  \Delta i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R)
  \]

- **Impurity measures**
  - **Entropy impurity (information gain):**
    \[
    i(N) = -\sum_j p(C_j|N) \log_2 p(C_j|N)
    \]
  - **Gini impurity:**
    \[
    i(N) = \sum_{i \neq j} p(C_i|N)p(C_j|N) = \frac{1}{2} \left[ 1 - \sum_j p^2(C_j|N) \right]
    \]

Recap: Overfitting Prevention (Pruning)

- Two basic approaches for decision trees
  - **Prepruning**: Stop growing tree as some point during top-down construction when there is no longer sufficient data to make reliable decisions.
    - Cross-validation
    - Chi-square test
    - MDL
  - **Postpruning**: Grow the full tree, then remove subtrees that do not have sufficient evidence.
    - Merging nodes
    - Rule-based pruning

- In practice often preferable to apply post-pruning.
Recap: ID3 Algorithm

- **ID3 (Quinlan 1986)**
  - One of the first widely used decision tree algorithms.
  - Intended to be used with nominal (unordered) variables
    - Real variables are first binned into discrete intervals.
  - General branching factor
    - Use gain ratio impurity based on entropy (information gain) criterion.

- **Algorithm**
  - Select attribute $a$ that best classifies examples, assign it to root.
  - For each possible value $v_i$ of $a$,
    - Add new tree branch corresponding to test $a = v_i$.
    - If example_list($v_i$) is empty, add leaf node with most common label in example_list($a$).
    - Else, recursively call ID3 for the subtree with attributes $A \setminus a$. 
Recap: C4.5 Algorithm

- **C4.5 (Quinlan 1993)**
  - Improved version with extended capabilities.
  - Ability to deal with real-valued variables.
  - Multiway splits are used with nominal data
    - Using gain ratio impurity based on entropy (information gain) criterion.
  - Heuristics for pruning based on statistical significance of splits.
  - Rule post-pruning

- **Main difference to CART**
  - Strategy for handling missing attributes.
  - When missing feature is queried, C4.5 follows all $B$ possible answers.
  - Decision is made based on all $B$ possible outcomes, weighted by decision probabilities at node $N$. 

B. Leibe
Recap: Computational Complexity

- **Given**
  - Data points \( \{x_1, \ldots, x_N\} \)
  - 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.

B. Leibe
Course Outline

• Fundamentals
  ➢ Bayes Decision Theory
  ➢ Probability Density Estimation

• Discriminative Approaches
  ➢ Linear Discriminant Functions
  ➢ Statistical Learning Theory
  ➢ Support Vector Machines
  ➢ Boosting, Decision Trees, Random Forests

• Generative Models
  ➢ Bayesian Networks
  ➢ Markov Random Fields

• Unifying Perspective
  ➢ HMMs, EM, CRFs, ...
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):

\[
\Delta E = \sum_{k=1}^{K} \frac{|S_k|}{|S|} \sum_{j=1}^{N} p_j \log_2(p_j)
\]
Recap: Ensemble Combination

- **Ensemble combination**
  - Tree leaves $(l, \eta)$ store posterior probabilities of the target classes.
  - Combine the output of several trees by averaging their posteriors (Bayesian model combination)

\[
p(C|\mathbf{x}) = \frac{1}{L} \sum_{l=1}^{L} p_{l,\eta}(C|\mathbf{x})
\]
Recap: Random Forests (Breiman 2001)

- 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...
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!
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.

```
+---+---+---+---+
|   |   |   |   |
| f0| f1| f2| f2 |
+---+---+---+---+
```

0 0 \rightarrow Update leaf $100_2 = 4$
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:
    \[
    p(f_1, \ldots, f_{N_f} | C_k) \approx \prod_{j=1}^{M} p(F_j | C_k) = p(f_1, \ldots, f_S | C_k) \cdot p(f_{S+1}, \ldots, f_{2S} | C_k) \cdot \ldots
    \]
    - Full joint inside fern
    - Naïve Bayes between ferns

$\Rightarrow$ Model with $M \cdot 2^S$ parameters (“Semi-Naïve”).
$\Rightarrow$ Flexible solution that allows complexity/performance tuning.
Course Outline

- Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation

- Discriminative Approaches
  - Linear Discriminant Functions
  - Statistical Learning Theory & SVMs
  - Boosting, Decision Trees, Random Forests
  - Model Selection

- Generative Models
  - Bayesian Networks
  - Markov Random Fields

- Unifying Perspective
  - HMMs, EM, CRFs, ...
Recap: Model Selection

- We’ve now seen even more different classifiers...
  - k-NN
  - Bayes classifiers
  - Fisher’s Linear Discriminant
  - SVMs
  - AdaBoost
  - Decision Trees
  - Randomized Trees & Ferns

- So... which one of them is best?

B. Leibe
Recap: Which Classifier Is Best?

- Can we say something about classification performance in general?
  - Can we give performance guarantees for certain classifiers?
    - Yes, to some degree. But the bounds are often too loose.
  - Can we find a classifier or learning algorithm that is consistently better than all others?
    - No. There is no free lunch. ⇒ No Free Lunch Theorem
  - Can we at least find a set of features that is consistently better than all others?
    - No. There is no “best” set of features. ⇒ Ugly Duckling Theorem
  - How can we then empirically choose between different classification models?
    - Model selection problem
Recap: Bayes Error Rate

- **Bayes error rate**
  - Irreducible error rate under optimal classification based on the test data’s ground truth labels.
  - Upper bound on achievable classifier accuracy.
  
  - We know that the error of the 1-NN classifier is at most twice the Bayes error.
  - However, this doesn’t say how well other classifiers would perform on the same data...
  - In practice, the Bayes error rate is mainly of theoretical interest...
Recap: Statistical Learning Theory

- **Structural Risk Minimization**
  - Aims to quantify the structural risk of the employed classifier through the VC confidence.
  - Approximation formula
    \[ R(\alpha) \leq R_{emp}(\alpha) + \epsilon(N, p^*, h) \]
  - We know that the SVM tries to minimize the structural risk of the classifier by maximizing the margin.
  - But the guaranteed risk (bound on generalization error) of the above formula is still in most cases too loose...

B. Leibe
Recap: No Free Lunch Theorem - Results

• Theorem

- For any two learning algorithms $p_1(h \mid D)$ and $p_2(h \mid D)$, the following statements are true independent of the sampling distribution $p(x)$ and the number of training points $n$:

1. Uniformly averaged over all target functions $F$,
   $$\mathcal{E}_1(E \mid F, n) - \mathcal{E}_2(E \mid F, n) = 0$$

"No matter how clever we are at choosing a “good” learning algorithm, if all target functions are equally likely, it will not outperform random guessing."
Recap: No Free Lunch Theorem - Results

• Theorem

➢ For any two learning algorithms $p_1(h|D)$ and $p_2(h|D)$, the following statements are true independent of the sampling distribution $p(x)$ and the number of training points $n$:

1. Uniformly averaged over all target functions $F$,
   $$\mathcal{E}_1(E|F, n) - \mathcal{E}_2(E|F, n) = 0$$

2. For any fixed training set $D$, uniformly averaged over $F$,
   $$\mathcal{E}_1(E|F, D) - \mathcal{E}_2(E|F, D) = 0$$

“This even holds if we know the training set $D$.”
Recap: No Free Lunch Theorem - Results

• Theorem

  For any two learning algorithms $p_1(h | D)$ and $p_2(h | D)$, the following statements are true independent of the sampling distribution $p(x)$ and the number of training points $n$:

  3. Uniformly averaged over all priors $p(F)$,
     \[ \mathcal{E}_1(E|n) - \mathcal{E}_2(E|n) = 0 \]

  4. For any fixed training set $D$, uniformly averaged over $p(F)$,
     \[ \mathcal{E}_1(E|D) - \mathcal{E}_2(E|D) = 0 \]

“The same is true for non-uniform target function distributions (again even if we know the training set $D$).”
Recap: No Free Lunch Theorem - Implications

possible learning systems

impossible learning systems

problem space (not feature space)

Image source: Duda & Hart, 2001
Recap: Ugly Duckling Theorem

• Theorem

  ➢ Given that we use a finite set of predicates that enables us to distinguish any two patterns under consideration, the number of predicates shared by any two such patterns is constant and independent of the choice of those patterns.

  ⇒ If pattern similarity is based on the total number of predicates shared by two patterns, then any two patterns are “equally similar”.

  ➢ This also holds for continuous feature spaces.

  “There is no problem independent or privileged or “best” set of features or feature attributes.”
Recap: Ugly Duckling Theorem - Implications

- There is no problem independent best set of features.
  - Even the notion of similarity between patterns is fundamentally based on assumptions about the problem domain.
  - Those assumptions may or may not be correct.
Recap: Occam’s Razor

• Philosophical principle
  “In science, all other things being equal, simple theories are preferable to complex ones”
  William of Occam (1285-1347)

  ➢ Best scientific theory is the smallest one that explains all the facts.

• Interpretation
  ➢ Most learning models (“theories”) make errors.
  ➢ Those errors are like exceptions to the theory.
  ➢ Information embedded in exceptions must be included as part of the theory when simplicity is judged.

Slide adapted from Chris Bryant
Recap: Minimum Description Length Principle

- **MDL Principle**
  
  "The best theory for a body of data is one that minimizes the size of the theory plus the amount of information necessary to specify exceptions."

  \[ \Rightarrow \text{Recommends choosing the } h \text{ that minimizes } L(h) + L(D|h). \]

- **Connection to Maximum-Likelihood estimation**
  
  - **Aim of ML** is to maximize \( p(h|D) \).
  
  - **Bayes rule:**
    \[
    p(h|D) = \frac{p(h)p(D|h)}{p(D)}
    \]
  
  - The optimal hypothesis is thus
    \[
    h^* = \arg \min_h \left[ -\log_2 p(h) - \log_2 p(D|h) \right]
    \]
  
  - This connection gives some credence to MDL principle.
Recap: Applying MDL to Clustering

- **Recall:** Difficulty in k-means and EM clustering
  - How to select the number of clusters \( k \)?
  - Cannot compare LMS error \( \Rightarrow \) approaches zero for \( k \to N \).

- **Idea**
  - Search for the clustering that minimizes the description length.
  - Need to define
    - \( p(D|h) \): likelihood of the data points under the cluster model.
    - \( p(h) \): model prior \( \Rightarrow \) Choose here the cost for specifying the model:

\[
\log p(D|h) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k)
\]

\[
\log p(h) = \sum_{k=1}^{K} [\log \pi_k + \log \mu_k + \log \Sigma_k]
\]
Recap: Other Model Selection Criteria...

- **Akaike Information Criterion (AIC)**
  - Choose the model which maximizes
    \[ \log p(D|h_{ML}) - M \]
    where
    - \( p(D|h_{ML}) \) is the best-fit log-likelihood and
    - \( M \) is the number of adjustable parameters in the model.

- **Bayes Information Criterion (BIC)**
  - Choose the model which maximizes
    \[ \log p(D|h) \approx \log p(D|h_{MAP}) - \frac{1}{2} M \log N \]
    where
    - \( p(D|h_{MAP}) \) is the best maximum-a-posteriori model.
    - \( N \) is the number of data points.
    \[ \Rightarrow \text{BIC penalizes complex models more heavily than AIC.} \]
Course Outline

• Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation

• Discriminative Approaches
  - Lin. Discriminants, SVMs, Boosting

• Graphical Models
  - Bayesian Networks
  - Markov Random Fields
  - Exact Inference
  - Approximate Inference

• Unifying Perspective
  - HMMs, EM, CRFs, ...
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.
Recap: Directed Graphical Models

- **Chains of nodes:**

\[
p(a) \quad p(b|a) \quad p(c|b)
\]

- 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:

\[
p(a, b, c) = p(c|a, b)p(a, b) = p(c|b)p(b|a)p(a)
\]
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:

    \[
    p(a, b, c) = p(c|a, b)p(a, b) = p(c|a, b)p(a)p(b)
    \]
Recap: Factorization of the Joint Probability

- Computing the joint probability

\[ p(x_1, \ldots, x_7) = p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5) \]

General factorization

\[ p(x) = \prod_{k=1}^{K} p(x_k|pa_{x_k}) \]

We can directly read off the factorization of the joint from the network structure!
Recap: Factorized Representation

- Reduction of complexity
  - Joint probability of $n$ binary variables requires us to represent values by brute force
    \[ O(2^n) \text{ terms} \]
  - The factorized form obtained from the graphical model only requires
    \[ O(n \cdot 2^k) \text{ terms} \]
    - $k$: 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.*
Recap: Conditional Independence

- \( X \) is conditionally independent of \( Y \) given \( V \)
  - Definition: \( X \perp Y | V \iff p(X|Y, V) = p(X|V) \)
  - Also: \( X \perp Y | V \iff p(X, Y | V) = p(X | V) p(Y | V) \)
  - Special case: Marginal Independence
    \( X \perp Y \iff X \perp Y | \emptyset \iff p(X, Y) = p(X) p(Y) \)
  - Often, we are interested in conditional independence between sets of variables:
    \( \mathcal{X} \perp \mathcal{Y} | \mathcal{V} \iff \{X \perp Y | \mathcal{V}, \ \forall X \in \mathcal{X} \ \text{and} \ \forall Y \in \mathcal{Y}\} \)
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

• 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 $C$.”
Recap: “Bayes Ball” Algorithm

- Graph algorithm to compute d-separation
  - **Goal:** Get a ball from $X$ to $Y$ without being blocked by $V$.
  - 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 ($W \notin V$) passes through balls from parents, but also bounces back balls from children.
  - An observed node ($W \in V$) bounces back balls from parents, but blocks balls from children.
Recap: The Markov Blanket

- Markov blanket of a node $x_i$
  - Minimal set of nodes that isolates $x_i$ from the rest of the graph.
  - This comprises the set of
    - Parents,
    - Children, and
    - Co-parents of $x_i$.

This is what we have to watch out for!

B. Leibe

Image source: C. Bishop, 2006
Course Outline

• Fundamentals
  ➢ Bayes Decision Theory
  ➢ Probability Density Estimation

• Discriminative Approaches
  ➢ Lin. Discriminants, SVMs, Boosting

• Graphical Models
  ➢ Bayesian Networks
  ➢ Markov Random Fields
  ➢ Exact Inference
  ➢ Approximate Inference

• Unifying Perspective
  ➢ HMMs, EM, CRFs, …
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 $A \perp B \mid C$.
  - Simple Markov blanket:
Recap: Factorization in MRFs

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

\[ p(x) = \frac{1}{Z} \prod_C \psi_C(x_C) \]

  - The normalization constant \( Z \) is called the partition function.

\[ Z = \sum_x \prod_C \psi_C(x_C) \]

• 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 \( O(K^M) \) 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")

\[ \psi_C(x_C) = \exp\{ -E(x_C) \} \]

- with an energy function \( E \).

• Why is this convenient?
  - Joint distribution is the product of potentials \( \Rightarrow \) 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").
  \(\Rightarrow\) This process is called moralization. It results in the moral graph.
Recap: Conversion Algorithm

- General procedure to convert directed $\rightarrow$ undirected
  1. Add undirected links to marry the parents of each node.
  2. Drop the arrows on the original links $\Rightarrow$ 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

- **Discriminative Approaches**
  - Lin. Discriminants, SVMs, Boosting

- **Graphical Models**
  - Bayesian Networks
  - Markov Random Fields
  - Exact Inference
  - Approximate Inference

- **Unifying Perspective**
  - HMMs, EM, CRFs, ...
Recap: Factor Graphs

- **Joint probability**
  - Can be expressed as **product of factors**: \( p(x) = \frac{1}{Z} \prod_s f_s(x_s) \)
  - 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!

B. Leibe

Image source: C. Bishop, 2006
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.

$$p(x) \propto \prod_{s \in \text{ne}(x)} \mu_{f_s \rightarrow x}(x)$$

- Computational effort
  - Total number of messages = $2 \cdot$ number of graph edges.

Slide adapted from Chris Bishop
Recap: Sum-Product Algorithm

- Two kinds of messages
  - **Message from factor node to variable nodes:**
    - **Sum** of factor contributions
      \[
      \mu_{f_s \rightarrow x}(x) \equiv \sum_{X_s} F_s(x, X_s) = \sum_{X_s} f_s(x_s) \prod_{m \in \text{ne}(f_s) \setminus x} \mu_{x_m \rightarrow f_s}(x_m)
      \]
  - **Message from variable node to factor node:**
    - **Product** of incoming messages
      \[
      \mu_{x_m \rightarrow f_s}(x_m) \equiv \prod_{l \in \text{ne}(x_m) \setminus f_s} \mu_{f_l \rightarrow x_m}(x_m)
      \]

⇒ Simple propagation scheme.
Recap: Sum-Product from Leaves to Root

Message definitions:

\[
\mu_{f_s \rightarrow x}(x) \equiv \sum_{X_s} f_s(x_s) \prod_{m \in \text{ne}(f_s) \setminus x} \mu_{x_m \rightarrow f_s}(x_m)
\]

\[
\mu_{x_m \rightarrow f_s}(x_m) \equiv \prod_{l \in \text{ne}(x_m) \setminus f_s} \mu_{f_l \rightarrow x_m}(x_m)
\]

\[
\mu_{x \rightarrow f}(x) = 1 \quad \quad \mu_{f \rightarrow x}(x) = f(x)
\]
Recap: Sum-Product from Root to Leaves

Message definitions:

\[ \mu_{f_s \rightarrow x}(x) \equiv \sum_{X_s} f_s(x_s) \prod_{m \in \text{ne}(f_s) \setminus x} \mu_{x_m \rightarrow f_s}(x_m) \]

\[ \mu_{x_m \rightarrow f_s}(x_m) \equiv \prod_{l \in \text{ne}(x_m) \setminus f_s} \mu_{f_l \rightarrow x_m}(x_m) \]

\[ \mu_{x \rightarrow f}(x) = 1 \]

\[ \mu_{f \rightarrow x}(x) = f(x) \]
Recap: Max-Sum Algorithm

• Objective: an efficient algorithm for finding
  - Value $x^{\text{max}}$ that maximises $p(x)$;
  - Value of $p(x^{\text{max}})$.

  ⇒ Application of dynamic programming in graphical models.

• Key ideas
  - We are interested in the maximum value of the joint distribution
    $p(x^{\text{max}}) = \max_x p(x)$
    ⇒ Maximize the product $p(x)$.
  - For numerical reasons, use the logarithm.
    $\ln \left( \max_x p(x) \right) = \max_x \ln p(x)$.
    ⇒ Maximize the sum (of log-probabilities).
Recap: Max-Sum Algorithm

- Initialization (leaf nodes)
  \[ \mu_{x \rightarrow f}(x) = 0 \quad \mu_{f \rightarrow x}(x) = \ln f(x) \]

- Recursion
  - **Messages**
    \[ \mu_{f \rightarrow x}(x) = \max_{x_1, \ldots, x_M} \left[ \ln f(x, x_1, \ldots, x_M) + \sum_{m \in \text{ne}(f_x) \setminus x} \mu_{x_m \rightarrow f}(x_m) \right] \]
    \[ \mu_{x \rightarrow f}(x) = \sum_{l \in \text{ne}(x) \setminus f} \mu_{f_l \rightarrow x}(x) \]
  - For each node, keep a record of which values of the variables gave rise to the maximum state:
    \[ \phi(x) = \arg \max_{x_1, \ldots, x_M} \left[ \ln f(x, x_1, \ldots, x_M) + \sum_{m \in \text{ne}(f_x) \setminus x} \mu_{x_m \rightarrow f}(x_m) \right] \]

Slide adapted from Chris Bishop
Recap: Max-Sum Algorithm

- Termination (root node)
  - Score of maximal configuration
    \[ p^{\text{max}} = \max_x \left[ \sum_{s \in \text{ne}(x)} \mu_{f_s \rightarrow x}(x) \right] \]
  - Value of root node variable giving rise to that maximum
    \[ x^{\text{max}} = \arg \max_x \left[ \sum_{s \in \text{ne}(x)} \mu_{f_s \rightarrow x}(x) \right] \]
  - Back-track to get the remaining variable values
    \[ x_{n-1}^{\text{max}} = \phi(x_n^{\text{max}}) \]
Recap: Junction Tree Algorithm

• 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.
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
Recap: Junction Tree Example

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

• Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation

• Discriminative Approaches
  - Lin. Discriminants, SVMs, Boosting

• Graphical Models
  - Bayesian Networks & Applications
  - Markov Random Fields & Applications
  - Exact Inference
  - Approximate Inference

• Unifying Perspective
  - HMMs, EM, CRFs, ...
Recap: MRF Structure for Images

- **Basic structure**
  - **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”.

- **Two components**
  - **Observation model**
  - **Neighborhood relations**
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
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
  \[ g_{ij}(y) = e^{-\beta \|y_i - y_j\|^2} \]
  \[ \beta = 2 / \text{avg} \left( \|y_i - y_j\|^2 \right) \]
  - Discourages label changes except in places where there is also a large change in the observations.
Recap: Graph Cuts for Binary Problems

“expected” intensities of object and background $I^s$ and $I^t$ can be re-estimated

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

EM-style optimization

Slide credit: Yuri Boykov
Recap: s-t-Mincut Equivalent to Maxflow

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

Regional term

\[ E(L) = \sum_p E_p(L_p) + \sum_{pq \in N} E(L_p, L_q) \]

t-links

Boundary term

n-links

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

\[ E(L) \text{ can be minimized by } s-t \text{ graph cuts } \iff E(s,s) + E(t,t) \leq E(s,t) + E(t,s) \]

Submodularity ("convexity")

- Submodularity is the discrete equivalent to convexity.
  - Implies that every local energy minimum is a global minimum.
  - Solution will be globally optimal.
Recap: $\alpha$-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.
Recap: Simple Binary Image Denoising Model

- **MRF Structure**

  ![MRF Structure Diagram](image)

  Observation process

  “Smoothness constraints”

  - Smoothness term: fixed penalty $\beta$ if neighboring labels disagree.
  - Observation term: fixed penalty $\eta$ if label and observation disagree.

  \[
  E(x, y) = h \sum_i x_i + \beta \sum_{i,j} \delta(x_i \neq x_j) + \eta \sum_i \delta(x_i \neq y_i)
  \]

  \(x_i, y_i \in \{0, 1\}\)

  Image source: C. Bishop, 2006
Recap: Converting an MRF into an s-t Graph

- Conversion:

\[ L_i = 1 \]
\[ h + \eta(1 - y_i) \]

- Energy:

\[ E(x, y) = h \sum_i x_i + \beta \sum_{\{i,j\}} \delta(x_i \neq x_j) + \eta \sum_i \delta(x_i \neq y_i) \]

- Unary potentials are straightforward to set.
  - Just insert \( x_i = 1 \) and \( x_i = 0 \) into the unary terms above…
Recap: Converting an MRF into an s-t Graph

- Conversion:

  \[ E(x, y) = h \sum_i x_i + \beta \sum_{\{i,j\}} \delta(x_i \neq x_j) - \eta \sum_i x_i y_i \]

  - **Unary potentials** are straightforward to set.
  - **Pairwise potentials** are more tricky, since we don’t know \( x_i \)!
    - Trick: the pairwise energy only has an influence if \( x_i \neq x_j \).
    - (Only!) in this case, the cut will go through the edge \( \{x_i, x_j\} \).

- Energy:

  \[ L_i = 1 \]
  \[ h + \eta(1 - y_i) \]
  \[ L_i = 0 \]
  \[ \eta y_i \]

See Exercise 5.4
Course Outline

- Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation

- Discriminative Approaches
  - Lin. Discriminants, SVMs, Boosting

- Graphical Models
  - Bayesian Networks & Applications
  - Markov Random Fields & Applications
  - Exact Inference
  - Approximate Inference

- Unifying Perspective
  - HMMs, EM, CRFs, ...
Recap: Parameter Learning in BNs

- We need to specify two things:
  - **Structure** of Bayesian network (graph topology)
  - Parameters of each *conditional probability table* (CPT)

- It is possible to learn both from training data.
  - But learning structure is much harder than learning parameters.
  - Also, learning when some nodes are hidden is much harder than when everything is observable.

- Four cases:

<table>
<thead>
<tr>
<th>Structure</th>
<th>Observability</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Known</td>
<td>Full</td>
<td>Maximum Likelihood Estimation</td>
</tr>
<tr>
<td>Known</td>
<td>Partial</td>
<td>EM (or gradient ascent)</td>
</tr>
<tr>
<td>Unknown</td>
<td>Full</td>
<td>Search through model space</td>
</tr>
<tr>
<td>Unknown</td>
<td>Partial</td>
<td>EM + search through model space</td>
</tr>
</tbody>
</table>

B. Leibe
Recap: Learning with Known Structure

• **ML-Learning with complete data** (no hidden variables)
  - Log-likelihood decomposes into sum of functions of $\theta_i$.
  - Each $\theta_i$ can be optimized separately.
  - ML-solution: simply calculate frequencies.

• **ML-Learning with incomplete data** (hidden variables)
  - Iterative EM algorithm.
  - E-step: compute expected counts given previous settings $\theta^{(t)}$ of parameters $E[n_{i,j,k} | D, \theta^{(t)}]$.
  - M-step: re-estimate parameters $\theta$ using the expected counts.
    \[
    \theta^{(t+1)}_{i,j,k} \leftarrow \frac{E[n_{i,j,k} | D, \theta^{(t)}]}{\sum_{k'} E[n_{i,j,k'} | D, \theta^{(t)}]}
    \]
Recap: Unknown Structure

- **Goal**
  - Learn a directed acyclic graph (DAG) that best explains the data.

- **Constraints-based learning**
  - Use statistical tests of marginal and conditional independence.
  - Find the set of DAGs whose d-separation relations match the results of conditional independence tests.

- **Score-based learning**
  - Use a global score such as BIC (Bayes Information Criterion).
  - Find a structure that maximizes this score.
Course Outline

• Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation

• Discriminative Approaches
  - Lin. Discriminants, SVMs, Boosting

• Graphical Models
  - Bayesian Networks & Applications
  - Markov Random Fields & Applications
  - Exact Inference
  - Approximate Inference

• Unifying Perspective
  - HMMs, EM, CRFs, ...
Recap: Sampling Idea

- **Objective:**
  - Evaluate expectation of a function $f(z)$ w.r.t. a probability distribution $p(z)$.
  
  $$
  \mathbb{E}[f] = \int f(z) p(z) \, dz
  $$

- **Sampling idea**
  - Draw $L$ independent samples $z^{(l)}$ with $l = 1, \ldots, L$ from $p(z)$.
  - This allows the expectation to be approximated by a finite sum
  
  $$
  \hat{f} = \frac{1}{L} \sum_{l=1}^{L} f(z^l)
  $$

  - As long as the samples $z^{(l)}$ are drawn independently from $p(z)$, then
    
    $$
    \mathbb{E}[\hat{f}] = \mathbb{E}[f]
    $$
  
  ⇒ **Unbiased estimate, independent of the dimension of $z$!**

Slide adapted from Bernt Schiele

B. Leibe

Image source: C.M. Bishop, 2006
Recap: Sampling from a pdf

- In general, assume we are given the pdf \( p(x) \) and the corresponding cumulative distribution:

\[
F(x) = \int_{-\infty}^{x} p(z) dz
\]

- To draw samples from this pdf, we can invert the cumulative distribution function:

\[
u \sim Uniform(0, 1) \Rightarrow F^{-1}(u) \sim p(x)
\]
Recap: Rejection Sampling

- **Assumptions**
  - Sampling directly from \( p(z) \) is difficult.
  - But we can easily evaluate \( p(z) \) (up to some norm. factor \( Z_p \)):
    \[
p(z) = \frac{1}{Z_p} \tilde{p}(z)
    \]

- **Idea**
  - We need some simpler distribution \( q(z) \) (called proposal distribution) from which we can draw samples.
  - Choose a constant \( k \) such that: \( \forall z : kq(z) \geq \tilde{p}(z) \)

- **Sampling procedure**
  - Generate a number \( z_0 \) from \( q(z) \).
  - Generate a number \( u_0 \) from the uniform distribution over \([0, kq(z_0)]\).
  - If \( u_0 > \tilde{p}(z_0) \) reject sample, otherwise accept.
Recap: Importance Sampling

• Approach
  - Approximate expectations directly (but does not enable to draw samples from $p(z)$ directly).
  - Goal:
    \[
    \mathbb{E}[f] = \int f(z)p(z)\,dz
    \]

• Idea
  - Use a proposal distribution $q(z)$ from which it is easy to sample.
  - Express expectations in the form of a finite sum over samples \(\{z^{(l)}\}\) drawn from $q(z)$.
    \[
    \mathbb{E}[f] = \int f(z)p(z)\,dz = \int f(z)\frac{p(z)}{q(z)}q(z)\,dz \\
    \approx \frac{1}{L} \sum_{l=1}^{L} \frac{p(z^{(l)})}{q(z^{(l)})} f(z^{(l)})
    \]

Image source: C.M. Bishop, 2006
Recap: MCMC - Markov Chain Monte Carlo

- **Overview**
  - Allows to sample from a large class of distributions.
  - Scales well with the dimensionality of the sample space.

- **Idea**
  - We maintain a record of the current state $z^{(\tau)}$.
  - The proposal distribution depends on the current state: $q(z | z^{(\tau)})$.
  - The sequence of samples forms a Markov chain $z^{(1)}, z^{(2)}, ...$

- **Approach**
  - At each time step, we generate a candidate sample from the proposal distribution and accept the sample according to a criterion.
  - Different variants of MCMC for different criteria.
Recap: MCMC - Metropolis Algorithm

- **Metropolis algorithm**
  - Proposal distribution is symmetric: \( q(z_A | z_B) = q(z_B | z_A) \)
  - The new candidate sample \( z^* \) is accepted with probability
    \[
    A(z^*, z^{(\tau)}) = \min \left( 1, \frac{\tilde{p}(z^*)}{\tilde{p}(z^{(\tau)})} \right)
    \]
  - New candidate samples always accepted if \( \tilde{p}(z^*) \geq \tilde{p}(z^{(\tau)}) \).
  - The algorithm sometimes accepts a state with lower probability.

- **Metropolis-Hastings Algorithm**
  - Generalization: Proposal distribution not necessarily symmetric.
  - The new candidate sample \( z^* \) is accepted with probability
    \[
    A(z^*, z^{(\tau)}) = \min \left( 1, \frac{\tilde{p}(z^*)q_k(z^{(\tau)} | z^*)}{\tilde{p}(z^{(\tau)})q_k(z^* | z^{(\tau)})} \right)
    \]
  - where \( k \) labels the members of the set of considered transitions.
Recap: Gibbs Sampling

• Approach
  - MCMC-algorithm that is simple and widely applicable.
  - May be seen as a special case of Metropolis-Hastings.

• Idea
  - Sample variable-wise: replace $z_i$ by a value drawn from the distribution $p(z_i | z_{\setminus i})$.
    - This means we update one coordinate at a time.
  - Repeat procedure either by cycling through all variables or by choosing the next variable.

• Properties
  - The algorithm always accepts!
  - Completely parameter free.
  - Can also be applied to subsets of variables.
Course Outline

• Fundamentals
  ➢ Bayes Decision Theory
  ➢ Probability Density Estimation

• Discriminative Approaches
  ➢ Lin. Discriminants, SVMs, Boosting

• Graphical Models
  ➢ Bayesian Networks & Applications
  ➢ Markov Random Fields & Applications
  ➢ Exact Inference
  ➢ Approximate Inference

• Unifying Perspective
  ➢ HMMs, EM, CRFs, ...
Recap: Markov Models

- **Markov assumption**
  - Each observation only depends on the most recent previous observation:
  
  \[ p(x_1, \ldots, x_N) = p(x_1) \prod_{n=2}^{N} p(x_n | x_{n-1}) \]

- **First-order Markov chain:**

- **Second-order Markov chain:**
Recap: Hidden Markov Models (HMMs)

- **Traditional view**
  - State-space model with \( k \) states, transition probabilities \( A \), and prob. distribution over output symbols \( \phi \).

- **Graphical Model view**
  - Latent variables \( Z \) for the current system state.
  - Observations \( X \) conditioned on \( Z \).
Three Main Tasks in HMMs

1. Likelihood Estimation
   - Given: an observation sequence and a model
   - What is the likelihood of this sequence given the model?
   "Forward-backward algorithm"
   \[ \Rightarrow \text{special case of Sum-Product!} \]

2. Finding the most probable state sequence
   - Given: an observation sequence and a model
   - What is the most likely sequence of states?
   "Viterbi algorithm"
   \[ \Rightarrow \text{special case of Max-Sum!} \]

3. Learning HMMs
   - Given: several observation sequences
   - How can we learn/adapt the model parameters?
   "Baum-Welch algorithm"
   application of EM!
Recap: 1. Likelihood Estimation in HMMs

- **Problem definition**
  - Given: HMM model $\theta = (A, \phi, \pi)$
  - sequence of observations $X = \{x_1, \ldots, x_T\}$
  - Goal: Compute the likelihood $p(X | \theta)$

- **Idea: Use factorization property**
  - Divide computation into a forward and a backward factor.

$$p(X | \theta) = \sum_{z_n} p(x_1, \ldots, x_N, z_n)$$

$$= \sum_{z_n} p(x_1, \ldots, x_n, z_n)p(x_{n+1}, \ldots, x_N | z_n)$$

$$= \sum_{z_n} \alpha(z_n) \beta(z_n)$$

B. Leibe
Recap: 1. Likelihood Estimation in HMMs

- **Recursive computation**
  - **Forward pass:**
    \[ \alpha(z_n) = p(x_n|z_n) \sum_{z_{n-1}} p(z_n|z_{n-1}) \alpha(z_{n-1}) \]
    \[ \alpha(z_1) = \pi p(x_1|\phi) \]
  - **Backward pass:**
    \[ \beta(z_n) = \sum_{z_{n+1}} p(x_{n+1}|z_{n+1}) p(z_{n+1}|z_n) \beta(z_{n+1}) \]
    \[ \beta(z_N) = 1 \]

- **Evaluate the likelihood**
  \[ p(X|\theta) = \sum_{z_n} \alpha(z_n) \beta(z_n) \]
  \[ = \sum_{z_N} \alpha(z_N) \]

B. Leibe
Recap: 2. Most Probable State Sequence

- Problem definition
  - Given: HMM model $\theta = (A, \phi, \pi)$
    - sequence of observations $X = \{x_1, \ldots, x_T\}$
  - Goal: Compute the most probable sequence of states
    
    $$w(Z) = \max_Z p(X, Z) = \max_{z_1, \ldots, z_N} p(x_1, \ldots, x_N, z_1, \ldots, z_N)$$

- Wait... Doesn’t this also look familiar?
  - Yes! We can simply apply the Max-Sum algorithm here.
Recap: 3. Learning the Model Parameters

- Problem definition
  - Given: data set $X = \{x_1, \ldots, x_N\}$
  - Goal: Determine the parameters of the HMM $\theta = (A, \phi, \pi)$ using maximum likelihood

- Idea: Marginalize over latent variables
  \[ p(X|\theta) = \sum_Z p(X, Z|\theta) \]

- Problems
  - The joint $p(X, Z|\theta)$ does not factorize over $n$.
  \[ \Rightarrow \text{We can’t treat each of summations over } z_n \text{ independently.} \]
  - We also can’t perform the summations explicitly ($K^N$ terms).
Recap: General EM Algorithm

• Algorithm
  1. Choose an initial setting for the parameters $\theta^{\text{old}}$.
  2. E step: Evaluate $p(Z|X, \theta^{\text{old}})$.
  3. M step: Evaluate $\theta^{\text{new}}$ given by
     \[ \theta^{\text{new}} = \arg \max_\theta Q(\theta, \theta^{\text{old}}) \]
     where
     \[ Q(\theta, \theta^{\text{old}}) = \sum_Z p(Z|X, \theta^{\text{old}}) \ln p(X, Z|\theta) \]
  4. Check for convergence of either the log likelihood or the parameter values. Until the criterion is satisfied, let
     \[ \theta^{\text{old}} \leftarrow \theta^{\text{new}} \]
     and return to step 2.

B. Leibe
Recap: 3. Learning the Model Parameters

• **E-Step:** Compute

\[
Q(\theta, \theta^{\text{old}}) = \sum_Z p(Z|X, \theta^{\text{old}}) \ln p(X, Z|\theta)
\]

• **M-Step:** Maximize \(Q(\theta, \theta^{\text{old}})\) w.r.t. \(\theta\)

  ➢ Solution (Bishop Ch.13)

\[
\pi_k = \frac{\gamma(z_{1k})}{\sum_{j=1}^{K} \gamma(z_{1j})}
\]

\[
A_{jk} = \frac{\sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nk})}{\sum_{l=1}^{K} \sum_{n=2}^{N} \xi(z_{n-1,l}, z_{nl})}
\]

\[
\phi_k = \ldots
\]

  (depending on the distribution \(\phi\))

➢ Important part: need to compute \(\gamma(z_n)\) and \(\xi(z_{n-1}, z_n)\) efficiently!

B. Leibe
Recap: 3. Learning the Model Parameters

- Efficient computation (using forward-backward algo.)

\[
\gamma(z_n) = \frac{\alpha(z_n)\beta(z_n)}{p(X)} \quad \text{with} \quad p(X) = \sum_{z_N} \alpha(z_N)
\]

\[
\xi(z_{n-1}, z_n) = \frac{\alpha(z_{n-1})p(x_n|z_n)p(z_n|z_{n-1})\beta(z_n)}{p(X)}
\]

- Interpretation
Any Questions?

So what can you do with all of this?
Mobile Object Detection & Tracking

[Ess, Leibe, Schindler, Van Gool, CVPR’08]
Topic: On-Line Learning of Appearance Models

• Goal: Data association in tracking
• Learn discriminative appearance model online
  ➢ Collect training samples online from tracklets using spatio-temporal constraints.
  ➢ Learn the appearance model using online Boosting.

Supervisor: Dennis Mitzel
(mitzel@umic.rwth-aachen.de)

[B. Leibe]
[C. Kuo, C. Huang R. Nevatia, CVPR'10]
Estimation of Human Pose and Motion using Nonparametric Belief Propagation

**Requirements**
1. Strong mathematical background
2. Knowledge in Computer Vision
3. Knowledge in Machine Learning
4. Programming experience in C++/Matlab

**Contact Information**
Georgios Floros  
UMIC, room 127  
Tel: + 49 (0) 241 80 20768  
E-mail: floros@umic.rth-aachen.de

Image Credit: Leonid Sigal
Mining the World’s Images...
Topics: Co-Segmentation for Object Discovery

- **Goal:** Discover small objects in millions of photos
- **Which pixels are shared by two images?**
  - Rough estimate: Homography / epipolar geometry inliers
  - Efficient solution using Markov Random Fields

Supervisor: Tobias Weyand
(weyand@umic.rwth-aachen.de)
Topic: Online Learning for Visual Inspection

- **External Master/Diplom thesis**
  - Automatic visual defect detection in MEMS production

- **Thesis focus**
  - Incremental learning
  - Optimize training time and defect detection capabilities

- **Details**
  - Work will be performed together with an industry partner in Grenoble, France
  - Contact: Bastian Leibe (leibe@umic.rwth-aachen.de)
Any More Questions?

Good luck for the exam!