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
- Optimal decision rule
  - Decide for $C_i$ 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)
    \[ \frac{p(x|C_1)}{p(C_1)} > \frac{p(x|C_2)}{p(C_2)} \]
- Decision regions: $R_1$, $R_2$, $R_3$ ...

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
  \[ E[L] = \sum_k \sum_j \int_{R_j} L_{kj}(x)p(x,C_k)dx \]
  - This can be done by choosing the regions $R_j$ such that
    \[ E[L] = \sum_k L_{kj}(x)p(C_k|x) \]
    which is easy to do once we know the posterior class probabilities $p(C_k|x)$. 

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
  \[ L_{\text{cancer diagnosis}} = \begin{pmatrix} 0 & 1000 \\ 1 & 0 \end{pmatrix} \]
Recap: The Reject Option

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

Recap: Gaussian (or Normal) Distribution

- One-dimensional case
  - Mean \( \mu \)
  - Variance \( \sigma^2 \)
  \[
  N(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left\{ -\frac{(x-\mu)^2}{2\sigma^2} \right\}
  \]

- Multi-dimensional case
  - Mean \( \mu \)
  - Covariance \( \Sigma \)
  \[
  N(x|\mu,\Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu) \right\}
  \]

Recap: Bayesian Learning Approach

- Bayesian view:
  - Consider the parameter vector \( \theta \) as a random variable.
  - When estimating the parameters, what we compute is
    \[
    p(x|\theta) = \int p(x, \theta|X) d\theta
    \]
  - Assumption: given \( \theta \), this doesn’t depend on \( X \) anymore.
  - \( p(x, \theta|X) = p(x|\theta, X)p(\theta|X) \)
  - \( p(x|\theta) = \int p(x|\theta)p(\theta|X) d\theta \)
  
  This is entirely determined by the parameter \( \theta \) (i.e. by the parametric form of the pdf).

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 (i.e. minimize the negative log-likelihood)
    - Take the derivative and set it to zero.
    \[
    \frac{\partial}{\partial \theta} E(\theta) = -\sum_{n=1}^{N} \frac{\partial}{\partial \theta} p(x_n|\theta) = 0
    \]

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: 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 \( D \).

Course Outline

- Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation
  - Mixture Models and EM
- Discriminative Approaches
  - Lin. Discriminants, SVM, Boosting
- Generative Models
  - Bayesian Networks
  - Markov Random Fields
  - Exact & Approximate Inference
- Unifying Perspective
  - HMMs, EM, CRFs, ...

Recap: Kernel Density Estimation

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

Recap: Mixture of Gaussians (MoG)

- "Generative model"
  \[ p(j) = \pi_j \]
  \[ p(x|j) \]
  \[ p(x) = \sum_{j=1}^{K} p(x|j)p(j) \]

Recap: MoG - Iterative Strategy

- Assuming we knew the values of the hidden variable...

\[ f(x) \]

- ML for Gaussian #1
  - \( h(j=1|x_n) \)
  - \( h(j=2|x_n) \)

- ML for Gaussian #2
  - \( \mu_1 = \sum_{n=1}^{N} h(j=1|x_n)x_n \)
  - \( \mu_2 = \sum_{n=1}^{N} h(j=2|x_n)x_n \)

- Bayes decision rule: Decide \( j = 1 \) if
  \[ p(j=1|x_n) > p(j=2|x_n) \]

- Assumining we knew the mixture components...

\[ f(x) \]

- \( p(j=1|x) \)
  - \( p(j=2|x) \)
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

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

Recap: Linear Discriminant Functions

- Basic idea
  - Directly encode decision boundary
  - Minimize misclassification probability directly.
- Linear discriminant functions
  \[ y(x) = w^T x + w_0 \]
  - \( w, w_0 \) 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
    \[ E(w) = \sum_{n=1}^{N} (y(x_n; w) - t_n)^2 \]
    \[ E_D(W) = \frac{1}{N} \text{Tr} \left\{ (XW - T)^T(XW - T) \right\} \]
  - Setting the derivative to zero yields
    \[ \bar{W} = (X^T X)^{-1} X^T T = \bar{X}^T T \]
  - We then obtain the discriminant function as
    \[ y(x) = \bar{W}^T x = T^T (\bar{X})^T \tilde{x} \]
  - 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
  \[ y(x) = g(w^T x + w_0) \]
  - \( g(\cdot) \) 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.

Recap: Extension to Nonlinear Basis Fcts.

- Generalization
  - Transform vector \( x \) with \( M \) nonlinear basis functions \( \phi_j(x) \):
    \[ y_k(x) = \sum_{j=1}^{M} w_j \phi_j(x) + w_k0 \]
- Advantages
  - Transformation allows non-linear decision boundaries.
  - By choosing the right \( \phi_j \), every continuous function can (in principle) be approximated with arbitrary accuracy.
- Disadvantages
  - The error function can in general no longer be minimized in closed form.
  \[ \Rightarrow \text{Minimization with Gradient Descent} \]

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.
  \[ \Rightarrow \text{Overfitting to the training set} \]

Recap: Linear Separability

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

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: Risk

- Empirical risk
  - Measured on the training/validation set
    \[ R_{\text{emp}}(\alpha) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i; \alpha)) \]
- Actual risk \( \equiv \) Expected risk
  - Expectation of the error on all data.
    \[ R(\alpha) = \int L(y, f(x; \alpha)) dP_{X, Y}(x, y) \]
  - \( P_{X, Y}(x, y) \) is the probability distribution of \( (x, y) \).
  - It is fixed, but typically unknown.
  - 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
    \[ R(\alpha) \leq R_{\text{emp}}(\alpha) + \epsilon(N, p^*, h) \]
  - 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 \( f \) points can be labeled in all possible \( 2^f \) 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
    \[ R(\alpha) \leq R_{\text{emp}}(\alpha) + \sqrt{\frac{h\log\left(\frac{2N}{h}\right) + 1 - \log\left(\frac{\eta}{4}\right)}{N}} \]
  - This bound is independent of \( P_{x,y}(x,y)! \)
  - If we know \( h \) (the VC dimension), we can easily compute the risk bound
    \[ R(\alpha) \leq R_{\text{emp}}(\alpha) + \epsilon(N, p^*, h) \]

Recap: Structural Risk Minimization

- How can we implement Structural Risk Minimization?
  \[ R(\alpha) \leq R_{\text{emp}}(\alpha) + \epsilon(N, p^*, h) \]
- **Classic approach**
  - Keep \( \epsilon(N, p^*, h) \) constant and minimize \( R_{\text{emp}}(\alpha) \).
  - \( \epsilon(N, p^*, h) \) can be kept constant by controlling the model parameters.
- **Support Vector Machines (SVMs)**
  - Keep \( R_{\text{emp}}(\alpha) \) constant and minimize \( \epsilon(N, p^*, h) \).
  - In fact: \( R_{\text{emp}}(\alpha) = 0 \) for separable data.
  - Control \( \epsilon(N, p^*, h) \) by adapting the VC dimension (controlling the “capacity” of the classifier).

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
    \[ f_n(w^T x_n + b) \geq 1 \quad \forall n \]
  - based on training data points \( x_n \) and target values \( f_n \in \{-1,1\} \).
Recap: SVM - Primal Formulation

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

- The solution of \( L_p \) needs to fulfill the KKT conditions
  - Necessary and sufficient conditions
    \[ a_n \geq 0 \]
    \[ \lambda \geq 0 \]
    \[ t_n y_n(x_n) - 1 \geq 0 \]
    \[ f(x) \geq 0 \]
    \[ a_n \{ y_n(x_n) - 1 \} = 0 \]
    \[ \lambda f(x) = 0 \]

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: \( a_n = 0 \) only for some points, the support vectors
  - Only the SVs actually influence the decision boundary!
  - Compute \( b \) by averaging over all support vectors:
    \[ b = \frac{1}{N_S} \sum_{n=1}^{N} \left( t_n - \sum_{m \in S} a_m t_m x_n^T x_n \right) \]

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 the hyperplane.
  - All other data points can be discarded!

Recap: SVM for Non-Separable Data

- Slack variables
  - One slack variable \( \xi \geq 0 \) for each training data point.
- Interpretation
  - \( \xi = 0 \) for points that are on the correct side of the margin.
  - \( \xi = y(x) - y(x) \) for all other points.

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

Recap: SVM - Dual Formulation

- Maximize
  \[ L_d(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (x_n^T x_m) \]
  under the conditions
  \[ a_n \geq 0 \]
  \[ \sum_{n=1}^{N} a_n t_n = 0 \]
- Comparison
  - \( L_d \) is equivalent to the primal form \( L_p \) but only depends on \( a_n \).
  - \( L_d \) scales with \( O(D^2) \).
  - \( L_d \) scales with \( O(N^2) \) - in practice between \( O(N) \) and \( O(N^2) \).

Recap: SVM - New Dual Formulation

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

- 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:
  - \( \phi: x \rightarrow \phi(x) \)

Recap: Kernels Fulfilling Mercer’s Condition

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

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

- Hyperbolic tangent kernel
  \[ k(x, y) = \tanh(kx^T y + \delta) \text{ e.g. Sigmoid} \]

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

Recap: The Kernel Trick

- Important observation
  - \( \phi(x) \) only appears in the form of dot products \( \phi(x)^T \phi(y) \):
    \[ y(x) = w^T \phi(x) + b \]
    \[ = \sum_{n=1}^{N} a_n \phi(x_n)^T \phi(x) + b \]
  - 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: Nonlinear SVM - Dual Formulation

- SVM Dual: Maximize
  \[ L_d(a) = \sum_{n=1}^{N} a_n \sum_{m=1}^{N} a_m t_n t_m k(x_n, x_n) \]
  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.


- 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
  - $h_i(x)$: “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:
    $$ H(x) = \text{sign} \left( \sum_{m=1}^{L} \alpha_m h_m(x) \right) $$

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: 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: 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 - Algorithm

1. Initialization: Set $w^{(1)} = \frac{1}{N}$ 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
      $$ J_m = \sum_{n=1}^{N} w^{(m)}_n I(h_m(x) \neq t_n) $$
      where $I(A) = 1$, if $A$ is true
      $$ I(A) = 0 $$
      otherwise
   b) Estimate the weighted error of this classifier on $x$:
      $$ \epsilon_m = \frac{1}{2} \ln \left( \frac{1-\epsilon_m}{\epsilon_m} \right) $$
   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:
      $$ w^{(m+1)}_n = w^{(m)}_n \exp \left\{ \alpha_m I(h_m(x_n) \neq t_n) \right\} $$
      for $n = 1, \ldots, N$.

Recap: Comparing Error Functions

- Ideal misclassification error function
- "Hinge error" used in SVMs
- Exponential error function
- "Cross-entropy error" $E = -\sum_{n=1}^{N} \left[ t_n \ln y_n + (1 - t_n) \ln(1 - y_n) \right]$
  - Similar to exponential error for $z > 0$.
  - Only grows linearly with large negative values of $z$.
  - Make AdaBoost more robust by switching to "GentleBoost"
- "Hinge error" used in SVMs
- 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!

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, EHN, CRFs, ...
Recap: Decision Trees

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

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 possible outcomes, weighted by decision probabilities at node N.
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 \( N \) 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.

- **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!
      \( \Rightarrow \) Result of discrete and greedy learning procedure.
  - Expensive learning step
    - Mostly due to costly selection of optimal split.

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):
    \[
    \Delta E = \sum_{i=1}^{K} \frac{|S_i|}{|T|} \sum_{j=1}^{N} p_j \log_2(p_j)
    \]

Recap: Ensemble Combination

- **Ensemble combination**
  - Tree leaves \( (i,j) \) store posterior probabilities of the target classes.
    \( p_{i,j}(C|x) \)
  - Combine the output of several trees by averaging their posteriors (Bayesian model combination)
    \[
    p(C|x) = \frac{1}{T} \sum_{i=1}^{T} p_{i,j}(C|x)
    \]
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: 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 (Semi-Naïve Bayes Classifiers)

- 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 \( f_1, \ldots, f_{N_f} \) into a binary number.
  - Update the “fern leaf” corresponding to that number.
    - Full joint inside fern
    - Naïve Bayes between ferns

\[
p(f_1, \ldots, f_{N_f} | C_k) \approx \prod_{j=1}^{N_f} p(F_j | C_k)
\]

\[
p(f_1, \ldots, f_{N_f} | C_k) = p(f_1 | C_k) \cdot p(f_2 | C_k) \cdot \ldots
\]
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?

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
    \[ R_N(a) \leq R_{\text{emp}}(a) + \epsilon(N, \rho^*, h) \]

- 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...

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 \( F \):
     \[ \mathcal{E_1}(F,n) - \mathcal{E_2}(F,n) = 0 \]

"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: 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. \( \Rightarrow \text{No Free Lunch Theorem} \)
  - Can we at least find a set of features that is consistently better than all others?
    - No. There is no “best” set of features. \( \Rightarrow \text{Ugly Duckling Theorem} \)
  - How can we then empirically choose between different classification models?
    - Model selection problem

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: 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 \( F \):
     \[ \mathcal{E_1}(F,n) - \mathcal{E_2}(F,n) = 0 \]

"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: 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: 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?
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 \( F \),
     \[ E_1(E|F, n) - E_2(E|F, n) = 0 \]
  2. For any fixed training set \( D \), uniformly averaged over \( F \),
     \[ E_1(E|F, D) - E_2(E|F, D) = 0 \]

  "This even holds if we know the training set \( D \)."

Recap: No Free Lunch 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: 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: Occam’s Razor

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

- **Interpretation**
  - Best scientific theory is the smallest one that explains all the facts.
  - 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.
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."

  \[ I(h) = -\log p(h) - \sum_{x \in D} p(x|h) \log p(x|h) \]

  \[ I(h) \leq I(h_{MAP}) + \frac{1}{2} M \log N \]

  \[ I(h) \approx I(h_{MAP}) - \frac{1}{2} M \log N \]

- **Connection to Maximum-Likelihood estimation**
  
  - Aim of ML is to maximize \( p(h|D) \).
  - Bayes rule: \( p(h|D) = \frac{p(D|h)p(h)}{p(D)} \)
  - The optimal hypothesis is thus

    \[ h^* = \arg \max \left[ -\log_2 p(h) - \log_2 p(D|h) \right] \]

  - This connection gives some credence to MDL principle.

Recap: Directed Graphical Models

- **Recall:** Directed Graphical Models
  
  - Directed or undirected

- **Bayesian Networks**
  
  - Chain rule: \( p(x_1, x_2, x_3) = p(x_1)p(x_2|x_1)p(x_3|x_2) \)

- **Markov Random Fields**
  
  - Emphasis on distributional property

- **Dec. Trees, Random Forests, Model Sel.**

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 \( \delta \) approaches zero for \( k \to N \).

- **Idea**
  
  - Search for the clustering that minimizes the description length.
  - Need to define

    \[ -p(D|h): \text{likelihood of the data points under the cluster model.} \]

    \[ \delta \text{ E.g. for Gaussian Mixture Model:} \]

    \[ \log p(D|h) = \sum_{i=1}^{K} \sum_{k=1}^{N_i} \pi_{ik} \log \mathcal{N}(x_i; \mu_k, \Sigma_k) \]

    \[ -p(h): \text{model prior. Choose here the cost for specifying the model:} \]

    \[ \log p(h) = \sum_{k=1}^{K} \log \pi_k + \log \mu_k + \log \Sigma_k \]

Recap: Other Model Selection Criteria...

- **Akaike Information Criterion (AIC)**
  
  - Choose the model which maximizes

    \[ \log p(D|h_{MLE}) - M \]

    where

    \[ -p(D|h_{MLE}) \] 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

    \[ \log p(D|h_{MAP}) - \frac{1}{2} M \log N \]

    where

    \[ -p(D|h_{MAP}) \] is the best maximum-a-posteriori model.

    \( N \) is the number of data points.

- **BIC penalizes complex models more heavily than AIC.**

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:

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

  - 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:
  \[
  \begin{align*}
  p(a, b) & = p(a, b) p(a) p(b) \\
  p(c, b) & = p(c, b) p(a, b) p(a) \\
  p(c, a) & = p(c, a) p(b, a) p(b)
  \end{align*}
  \]

  - 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 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

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

General factorization

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

Recap: Factorized Representation

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

- 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.

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 \( A \perp B \mid C \).
  - 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 → Y without being blocked by Y
  - 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 \( (i' \in I) \) passes through balls from parents, but also bounces back balls from children.
  - An observed node \( (i' \in I) \) 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 \( x_i \).

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!

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:

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
- Need to introduce additional links ("marry the parents").
  ⇒ This process is called moralization. It results in the moral graph.
- Need a clique of \( x_{i-1}, x_i, x_{i+1} \) to represent this factor!
- Add undirected links to the original links ("marry the parents").
- 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 \( p(x_i) \).
- 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 \( x_i \).
- Define the messages recursively:
  \[
  \mu_i(x_i) = \sum_{x_{i-1}} \psi_{i-1,i}(x_{i-1}, x_i) \mu_{i-1}(x_{i-1}) \\
  \mu_j(x_j) = \sum_{x_{j+1}} \psi_{j,j+1}(x_j, x_{j+1}) \mu_{j+1}(x_{j+1})
  \]
- Compute the normalization constant \( Z \) at any node \( x_{i0} \):
  \[
  Z = \sum_{x_{i0}} \mu(x_{i0}) \psi(x_{i0})
  \]

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, Etc, CRFs,...
Recap: Factor Graphs

- Joint probability
  - Can be expressed as product of factors: \( p(x) = \frac{1}{Z} \prod f_i(x) \)
  - 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

- Two kinds of messages
  - Message from factor node to variable nodes:
    - Sum of factor contributions
    \[ \mu_{f_i \rightarrow x}(x) \equiv \sum_{X_{\backslash i}} f_i(x, X_i) \]
    - Message from variable node to factor node:
      - Product of incoming messages
      \[ \mu_{x \rightarrow f_i}(x_m) \equiv \prod_{m \in i(x)} \mu_{f_i \rightarrow x_m}(x_m) \]

\[ \Rightarrow \text{Simple propagation scheme}. \]

Recap: Sum-Product from Leaves to Root

Message definitions:

\[ \mu_{f_i \rightarrow x}(x) \equiv \sum_{X_{\backslash i}} f_i(x, X_i) \prod_{m \in i(x)} \mu_{f_i \rightarrow x_m}(x_m) \]

Recap: Sum-Product from Root to Leaves

Message definitions:

\[ \mu_{x \rightarrow f_i}(x_m) \equiv \prod_{l \in (f_i \backslash x_m)} \mu_{f_i \rightarrow x_l}(x_l) \]

Recap: Max-Sum Algorithm

- Objective: an efficient algorithm for finding
  - Value \( x^{\text{argmax}} \) that maximises \( p(x) \).
  - Value of \( p(x^{\text{argmax}}) \).
  - Application of dynamic programming in graphical models.

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

- Initialization (leaf nodes)
  \[ p_{\text{max}}(x) = 0 \quad p_{\text{max}}(x) = \ln f(x) \]
- Recursion
  \[ p_{\text{max}}(x) = \max_{\text{other variables}} \left[ \ln f(x_1, \ldots, x_n) + \sum_{x_{n+1}} p_{\text{max}}(x_{n+1}) \right] \]
  \[ p_{\text{max}}(x) = \sum_{\text{other variables}} p_{\text{max}}(x) \]
  For each node, keep a record of which values of the variables gave rise to the maximum state:
  \[ p_{\text{max}}(x) = \max_{\text{other variables}} \left[ \ln f(x_1, \ldots, x_n) + \sum_{x_{n+1}} p_{\text{max}}(x_{n+1}) \right] \]

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.

Recap: Junction Tree Example

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

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
  
  Noisy observations
  
  "True" image content

- Two components
  
  Observation model
  - How likely is it that node \( y_i \) has label \( L \), 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".

Recap: How to Set the Potentials?

- Unary potentials
  - E.g., color model, modeled with a Mixture of Gaussians
    \[
    \phi(x_i, y_i; \theta) = \log \sum_k \theta_k p_k(x_i) N(y_i; y_k, \Sigma_k)
    \]
  - Learn color distributions for each label

- Pairwise potentials
  - Potts Model
    \[
    \psi(x_i, x_j; \theta) = \theta_\delta(s_i \neq s_j)
    \]
  - 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"
    \[
    \psi(x_i, x_j, y_i; \theta) = \theta_\delta(s_i \neq s_j)
    \]
  - Discourages label changes except in places where there is also a large change in the observations.

Recap: Graph Cuts for Binary Problems

- Solution will be globally optimal.

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

- s-t graph cuts can only globally minimize binary energies that are submodular.
  - Submodularity is the discrete equivalent to convexity.
  - Implies that every local energy minimum is a global minimum.
  - 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.

Recap: Simple Binary Image Denoising Model

- MRF Structure
  - Observation process
    - "True" image content
    - "Smoothness constraints"
    - Example: simple energy function ("Potts model")
      \[ E(x, y) = h \sum x_i + \beta \sum_{i \neq j} \delta(x_i \neq x_j) + \eta \sum_{i \neq j} \delta(x_i \neq y_i) \]
  - Smoothness term: fixed penalty \( \beta \) if neighboring labels disagree.
  - Observation term: fixed penalty \( \eta \) if label and observation disagree.

Recap: Converting an MRF into an s-t Graph

- Conversion:
  - \( L_i = 1 \) \[ h + \eta(1 - y_i) \]
  - \( L_i = 0 \) \[ h + \eta y_i \]
- Energy:
  - 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:
  - \( L_i = 1 \) \[ h + \eta(1 - y_i) \]
  - \( L_i = 0 \) \[ h + \eta y_i \]
- Energy:
  - Unary potentials are straightforward to set.
  - Pairwise potentials are more tricky, since we don’t know \( x_j \)
    - 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 \( [x_i, x_j] \).

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:
  - Known Full: Maximum Likelihood Estimation
  - Known Partial: EM (or gradient ascent)
  - Unknown Full: Search through model space
  - Unknown Partial: EM + search through model space
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.

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.

Recap: Sampling Idea

- Objective:
  - Evaluate expectation of a function $f(x)$ w.r.t. a probability distribution $p(x)$.
  \[ E[f] = \int f(x)p(x)dx \]

- Sampling idea
  - Draw $l$ independent samples $x^{(i)}$ with $i = 1, \ldots, l$ from $p(x)$.
  - This allows the expectation to be approximated by a finite sum
  \[ \hat{f} = \frac{1}{l} \sum_{i=1}^{l} f(x^{(i)}) \]
  - As long as the samples $x^{(i)}$ are drawn independently from $p(x)$, then
  \[ E[\hat{f}] = E[f] \]

  **Unbiased estimate, independent of the dimension of $x$!**

Recap: Rejection Sampling

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

- **Idea**
  - We need some simpler distribution $g(x)$ (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 $u_0$ from $q(z)$.
  - Generate a number $u$, from the uniform distribution over $[0, kq(z)]$,
  - If $u_0 > \tilde{p}(z)$ reject sample, otherwise accept.

Recap: Sampling from a pdf

- In general, assume we are given the pdf $p(x)$ and the corresponding cumulative distribution:
  \[ F(z) = \int_{-\infty}^{z} p(x)dz \]

- To draw samples from this pdf, we can invert the cumulative distribution function:
  \[ u \sim \text{Uniform}(0,1) = F^{-1}(z) \sim p(x) \]
Recap: Importance Sampling

- **Approach**
  - Approximate expectations directly (but does not enable to draw samples from $p(x)$ directly).
  - Goal: $\mathbb{E}[f] = \int f(x)p(x)dx$
- **Idea**
  - Use a proposal distribution $q(x)$ from which it is easy to sample.
  - Express expectations in the form of a finite sum over samples $(x^{(i)})$ drawn from $q(x)$.
  - Approximate expectations directly:
  
  $\mathbb{E}[f] \approx \frac{1}{N} \sum_i f(x^{(i)}) q(x^{(i)}) p(x^{(i)})$

Recap: MCMC - Metropolis Algorithm

- **Metropolis algorithm**
  - Proposal distribution is symmetric: $q(x_1|x_2) = q(x_2|x_1)$
  - The new candidate sample $x'$ is accepted with probability
    $A(x^*, x^{(i)}) = \min \left( 1, \frac{p(x^*)}{p(x^{(i)})} \right)$
  - New candidate sample $x'$ always accepted if $p(x^*) \geq p(x^{(i)})$.
  - The algorithm sometimes accepts a state with lower probability.
- **Metropolis-Hastings Algorithm**
  - Generalization: Proposal distribution not necessarily symmetric.
  - The new candidate sample $x'$ is accepted with probability
    $A(x^*, x^{(i)}) = \min \left( 1, \frac{p(x^*) q(x^{(i)}|x^*)}{p(x^{(i)}) q(x^*|x^{(i)})} \right)$
  - where $\tau$ 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 $x_i$ by a value drawn from the distribution $p(x_i|x_{-i})$.
  - 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:
    $p(x_1, \ldots, x_t) = p(x_t|x_{t-1}) \prod_{i=1}^{t-1} p(x_i|x_{i-1})$
- **First-order Markov chain:**
  
  $X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow X_4$
- **Second-order Markov chain:**
  
  $X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow X_4$
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 \( Z \).

\[
\begin{align*}
\mathbf{X} & = \{x_1, \ldots, x_T\} \\
\mathbf{Z} & = \{z_1, \ldots, z_T\} \\
\mathbf{A} & = \{a_{ij}\} \\
\mathbf{\Phi} & = \{\phi_k\} \\
\mathbf{\Pi} & = \{\pi_i\}
\end{align*}
\]

Recap: 1. Likelihood Estimation in HMMs

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

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

\[
p(\mathbf{X} | \theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} | \theta) = \sum_{\mathbf{Z}} p(\mathbf{X} | \mathbf{Z}) p(\mathbf{Z} | \theta)
\]

Recap: 2. Most Probable State Sequence

- **Problem definition**
  - Given: HMM model \( \theta = (\mathbf{A}, \mathbf{\phi}, \mathbf{\Pi}) \)
  - Sequence of observations \( X = \{x_1, \ldots, x_T\} \)
  - Goal: Compute the most probable sequence of states

\[
w(\mathbf{Z}) = \max_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} | \theta) = \max_{\mathbf{Z}} \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} | \theta)
\]

- **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 \( X = \{x_1, \ldots, x_T\} \)
  - Goal: Determine the parameters of the HMM \( \theta = (\mathbf{A}, \mathbf{\phi}, \mathbf{\Pi}) \) using maximum likelihood

- **Idea:** Marginalize over latent variables

\[
p(\mathbf{X} | \theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} | \theta)
\]

- **Problems**
  - The joint \( p(\mathbf{X}, \mathbf{Z} | \theta) \) does not factorize over \( \mathbf{z}_i \).
  - We can’t treat each of summations over \( \mathbf{z}_i \) independently.
  - We also can’t perform the summations explicitly \( (2^T)^n \) terms.
Recap: General EM Algorithm

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

Recap: 3. Learning the Model Parameters

- Efficient computation (using forward-backward algo.)

  $$
  \gamma(z_n) = \frac{\alpha(z_n) \beta(z_n)}{p(X)}
  $$

  with

  $$
  p(X) = \sum_{z_N} \alpha(z_N)
  $$

  and

  $$
  \zeta(z_{n-1}, z_n) = \frac{\alpha(z_{n-1}) p(z_{n-1}|z_n) p(z_n|z_{n-1}) \beta(z_n)}{p(X)}
  $$

  **Interpretation**

  ![Interpretation Diagram]

Any Questions?

**So what can you do with all of this?**

**Mobile Object Detection & Tracking**

**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)
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.rwth-aachen.de

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!