B(E)3M33UI: Competencies

Petr Pošík

May 26, 2017

Abstract

List of competencies students shall gain after completing course *Artificial Intelligence* taught at Czech Technical University in Prague, Dept. of Cybernetics, primarilly for (but not limited to) students of Cybernetics and Robotics master study programme, branch Robotics.

1 Bayesian decision tasks. Non-bayesian tasks. Empirical learning.

After this lecture, a student shall be able to ...

- explain various views on AI and describe the differences of their personal view of AI;
- list the fields of science most related to AI;
- define Bayesian decision task and all its components (decision strategy, risk, penalty function, observation, hidden state, joint probability distribution);
- solve simple instances of Bayesian decision task by hand, write a computer program solving Bayesian decision tasks;
- explain features of Bayesian strategy;
- recognize special cases of Bayesian decision task (minimization of error probability when estimating hidden state, strategy with "dontknow" decision);
- describe reasons and examplify situations when the Bayesian approach cannot be used;
- define and describe examples of non-Bayesian tasks which can be solved to some extent without learning (Neyman-Pearson, minimax, Wald);
- solve simple instances of the above non-Bayesian decision tasks by hand, write a computer program solving them;
- define the decision strategy design as a learning from data;
- describe the differences between Bayesian decision tasks, non-Bayesian decision tasks and decision tasks solved by learning;
- define the types of learning (supervised, unsupervised, semisupervised, reinforcement) and describe conceptual differences between them;
- define classification and regression types of problems, recognize them in practical situations;
- describe 2 approaches to learning (as parameter estimation, as direct optimal strategy design) and give examples of surrogate criteria used in them.

2 Linear methods for regression and classification.

After this lecture, a student shall be able to ...

- define and recognize linear regression model (with scalar parameters, in scalar product form, in matrix form, non-homogenous and homogenous coordinates);
- define the loss function suitable for fitting a regression model;
- explain the least squares metod, draw an illustration;
- compute coefficients of simple (1D) linear regression by hand, write a computer program computing coefficients for multiple regression;
- explain the concept of discrimination function for binary and multinomial classification;

- define a loss function suitable for fitting a classification model;
- describe a perceptron algorithm, perform a few iterations by hand;
- explain the characteristics of the perceptron algorithm;
- describe logistic regression, the interpretation of its outputs, and why we classify it as a linear model;
- define loss functions suitable for fitting logistic regression;
- define optimal separating hyperplane, explain in what sense it is optimal;
- define what a margin is, what support vectors are, and explain their relation;
- compute the margin given the parameters of separating hyperplane for which $\min_{i:y^{(i)}=+1}(x^{(i)}w^T + w_0) = 1$ and $\max_{i:y^{(i)}=-1}(x^{(i)}w^T + w_0) = -1$;
- formulate the primary quadratic programming task which results in the optimal separating hyperplane (including the soft-margin version);
- compute the parameters of optimal hyperplane given the set of support vectors and their weights.

3 Non-linear models. Basis expansion. Overfitting. Regularization.

After this lecture, a student shall be able to ...

- explain the reason for doing basis expansion (feature space straightening), and describe its principle;
- show the effect of basis expansion with a linear model on a simple example for both classification and regression settings;
- implement user-defined basis expansions in certain programming language;
- list advantages and disadvantages of basis expansion;
- explain why the error measured on the training data is not a good estimate of the expected error of the model for new data, and whether it under- or overestimates the true error;
- explain basic methods to get unbiased estimate of the true model error (testing data, k-fold crossvalidation, LOO crossvalidation);
- describe the general form of the dependency of training and testing errors on the model complexity/flexibility/capacity;
- define overfitting;
- discuss high bias and high variance problems of models;
- explain how to proceed if a suitable model complexity must be chosen as part of the training process;
- list 2 basic methods for overfitting prevention;
- describe the principles of ridge (Tikhonov) and lasso regularizations and their effects on the model parameters.

4 Nearest neighbors. Kernels, SVM. Decision trees.

After this lecture, a student shall be able to ...

- explain, use, and implement the method of k nearest neighbors for both classification and regression;
- explain the influence of k on the form of the final model;
- describe advantages and disadvantages of k-NN, and suggest a way hot to find a suitable value of k;
- show how to force the algorithm for learning the optimal separating hyperplane to find a nonlinear model using basis expansion, and using a kernel function;
- explain the meaning of kernels, and their advantages compared to basis expansion;
- explain the principle of support vector machine;
- describe the structure of a classification and regression tree, and the way it is used to determine a prediction;
- know a lower bound on the number of Boolean decision trees for a dataset with n attributes;
- describe TDIDT algorithm and its features, and know whether it will find the optimal tree;

- explain how to choose the best attribute for a split, and be able to manually perform the choice for simple examples;
- describe 2 methods to prevent tree overfitting, and argue which of them is better;
- explain how a decision tree can handle missing data during training and during prediction;
- describe what happens in a tree-building algorithm and what to do if the dataset contains an attribute with unique value for each observation;
- explain how to handle continuous input and output variables (as opposed to the discrete attributes).

5 Bagging. Random forests. Boosting.

After this lecture, a student shall be able to ...

- describe the basic principle behind all committee/ensemble methods;
- list and conceptually compare several methods to achieve diversity among models trained on the same data, and know which of these methods are used in which ensemble algorithms;
- explain the purpose and the basic principle of stacking;
- explain how a bootstrap sample is created from the available data, and describe its properties;
- describe features of bagging;
- explain how to compute out-of-bag error estimate when using bagging;
- explain the principle of random forests and describe their difference to bagging with trees;
- explain how to compute a score of variable importance using random forest;
- explain the hypothesis boosting problem, and define a weak and a strong classifier in this context;
- explain the basic principle of AdaBoost.M1 algorithm;
- relate the training error of the AdaBoost algorithm to the number of constituent models and to the errors of individual models;
- describe the relations of AdaBoost.M1, L2Boost, and Gradient Boosting.

6 Bayesian networks.

After this lecture, a student shall be able to ...

- explain why the joint probability distribution is an awkward model of domains with many random variables;
- define what a Bayesian network is, and describe how it solves the issues with joint probability;
- explain how a BN factorizes the joint distribution, and compare it with the factorization we get from chain rule;
- write down factorization of the joint probability given the BN graph, and vice versa, draw the BN graph given a factorization of the joint probability;
- explain the relation between the direction of edges in BN and the causality;
- given the structure of a BN, check whether 2 variables are guaranteed to be independent using the concept of D-separation;
- describe and prove the conditional (in)dependence relations among variable triplets (causual chain, common cause, common effect);
- describe inference by enumeration and explain why it is unwieldy for BN;
- explain the difference between inference by enumeration and by variable elimination (VE);
- explain what makes VE more suitable for BN than enumeration;
- describe the features (complexity) of exact inference by enumeration and VE in BN;
- explain how we can use sampling to make approximate inference in BN;
- describe Gibbs sampling.

7 Hidden Markov models.

After this lecture, a student shall be able to ...

- define Markov Chain (MC), describe assumptions used in MCs;
- show the factorization of the joint probability distribution used by 1st-order MC;
- understand and implement the mini-forward algorithm for prediction;
- explain the notion of the stationary distribution of a MC, describe its features, compute it analytically for simple cases;
- define Hidden Markov Model (HMM), describe assumptions used in HMM;
- explain the factorization of the joint probability distribution of states and observations implied by HMM;
- define the main inference tasks related to HMMs;
- explain the principles of forward, forward-backward, and Viterbi algorithms, implement them, and know when to apply them;
- compute a few steps of the above algorithms by hand for simple cases;
- describe issues that can arise in practice when using the above algorithms.

8 Expectation maximization algorithm.

After this lecture, a student shall be able to ...

- define and explain the task of maximum likelihood estimation;
- explain why we can maximize log-likelihood instead of likelihood, describe the advantages;
- describe the issues we face when trying to maximize the likelihood in case of incomplete data;
- explain the general high-level principle of Expectation-Maximization algorithm;
- describe the pros and cons of the EM algorithm, especially what happens with the likelihood in one EM iteration;
- describe the EM algorithm for mixture distributions, including the notion of responsibilities;
- explain the Baum-Welch algorithm, i.e. the application of EM to HMM; what parameters are learned and how (conceptually).

9 Planning.

After this lecture, a student shall be able to ...

- define planning and scheduling;
- provide a formal definition of a planning domain and a planning problem;
- define a plan as a classical plan or policy;
- provide eight assumptions of classical AI planning;
- define the atomic and factored state representation and justify its usage in AI planning methods;
- define a planning operator and action in a STRIPS planning domain;
- define the conditions of an applicable action in a state;
- define the state transition function for an applicable action in a state;
- describe the forward state-space search algorithm;
- define a relevant action for a goal and its regression set;
- describe the backward state-space search algorithm;
- describe characteristics of the plan-space search;
- define a negative/positive threat to a causal link and its possible solutions;
- describe GRAPHPLAN principles;
- define the GRAPHPLAN mutex relation construction;

- list at least five classical planner applications;
- define HTN planning;
- explain differences between STN and HTN;
- describe the abstract HTN algorithm;
- list basic properties of the Pyhop planner.

10 Scheduling.

After this lecture, a student shall be able to ...

- determine a schedule;
- define a complete, a partial, a consistent, and an optimal schedule;
- define basic static and dynamic parameters of jobs;
- define Graham's classification of scheduling problems;
- define machine environments 1, Pm, Qm, Rm;
- define machine environments Fm, FFs, Jm, Om;
- define precedence constraints, setup time and cost;
- define optimization criteria given by makespan, lateness, job tardiness;
- describe the local search algorithm;
- describe the tabu search algorithm;
- compute a schedule using the tabu search algorithm for simple examples;
- describe Johnson's algorithm;
- compute a schedule using Johnson's algorithm;
- describe the critical path method algorithm;
- compute a schedule using the critical path method.

11 Constraint satisfaction problems.

After this lecture, a student shall be able to ...

- define a constraint satisfaction problem;
- list the types of constraints and their specifications;
- describe a constraint graph;
- describe a generate-and-test algorithm;
- describe the standard search formulation;
- describe the backtracking search;
- list four basic variable and value ordering heuristics;
- describe the forward checking method;
- describe the AC-3 algorithm;
- describe the algorithm for tree-structured CSPs;
- describe an iterative algorithm using min-conflicts heuristics.

12 Neural networks.

After this lecture, a student shall be able to ...

- describe the model of a simple neuron, and explain its relation to multivariate regression and logistic regrassion;
- explain how to find weights of a single neuron using gradient descent (GD) algorithm;
- derive the update equations used in GD to optimize the weights of a single neuron for various loss functions and various activation functions;
- describe a multilayer feedforward network and discuss its usage and characteristics;
- compare the use of GD in case of a single neuron and in case of NN, discuss similarities and differences;
- explain the error backpropagation (BP) algorithm — its purpose and principle;
- implement BP algorithm for a simple NN, and suggest how the implementation should be modified to allow application for complex networks;
- discuss the purpose of various modifications of GD algorithm (learning rate decay, weight update schedule, momentum, ...);
- discuss the regularization options for NN (weight decay, dropout);
- describe RBF networks and explain their main differences to MLP.

13 Deep learning.

After this lecture, a student shall be able to ...

- define what a deep learning is and how it is related to representation learning;
- describe word embeddings (word2vec) and exemplify its features;
- explain why deep networks are hard to train, especially the vanishing gradient effect;
- describe the factors that facilitated the practical use of deep learning in the last decade;
- explain what an autoencoder is and how it is related to pre-training of deep nets;
- explain the principle of convolutional layers, the importance of weight sharing and pooling, and describe their main differences from fully connected layers;
- give examples of applications of convolutional networks to image processing;
- describe the difference of recurrent neural networks from feedforward networks;
- describe the unrolling of RNN in time and give an example;
- explain the difference between short- and long-term dependencies when processing sequences;
- describe LSTM and its main differences from a regular RNN unit;
- explain what the backpropagation in time is and what it is used for;
- give examples of applications of recurrent neural networks to language processing.