

# 10-301/10-601 Learning Objectives

## Course Level Learning Outcomes

1. Course Level
  - a. Implement and analyze existing learning algorithms, including well-studied methods for classification, regression, structured prediction, clustering, and representation learning
  - b. Integrate multiple facets of practical machine learning in a single system: data preprocessing, learning, regularization and model selection
  - c. Describe the the formal properties of models and algorithms for learning and explain the practical implications of those results
  - d. Compare and contrast different paradigms for learning (supervised, unsupervised, etc.)
  - e. Design experiments to evaluate and compare different machine learning techniques on real-world problems
  - f. Employ probability, statistics, calculus, linear algebra, and optimization in order to develop new predictive models or learning methods
  - g. Given a description of a ML technique, analyze it to identify (1) the expressive power of the formalism; (2) the inductive bias implicit in the algorithm; (3) the size and complexity of the search space; (4) the computational properties of the algorithm; (5) any guarantees (or lack thereof) regarding termination, convergence, correctness, accuracy or generalization power.

## ML Basics

1. Course Overview / Decision Trees
  - a. Formulate a well-posed learning problem for a real-world task by identifying the task, performance measure, and training experience
  - b. Describe common learning paradigms in terms of the type of data available and when, the form of prediction, and the structure of the output prediction
  - c. Identify examples of the ethical responsibilities of an ML expert
2. Decision Trees / Information Theory
  - a. Formalize a learning problem by identifying the input space, output space, hypothesis space, and target function
  - b. Implement Decision Tree training and prediction
  - c. Use effective splitting criteria for Decision Trees and be able to define entropy, conditional entropy, and mutual information / information gain
  - d. Explain the difference between memorization and generalization [CIML]
  - e. Describe the inductive bias of a decision tree
  - f. Judge whether a decision tree is "underfitting" or "overfitting"
  - g. Explain the difference between true error and training error

- h. Implement a pruning or early stopping method to combat overfitting in Decision Tree learning
- 3. k-Nearest Neighbors
  - a. Describe a dataset as points in a high dimensional space [CIML]
  - b. Implement k-Nearest Neighbors with  $O(N)$  prediction
  - c. Describe the inductive bias of a k-NN classifier and relate it to feature scale [a la. CIML]
  - d. Sketch the decision boundary for a learning algorithm (compare k-NN and DT)
  - e. State Cover & Hart (1967)'s large sample analysis of a nearest neighbor classifier
  - f. Invent "new" k-NN learning algorithms capable of dealing with even k
- 4. Model Selection
  - a. Plan an experiment that uses training, validation, and test datasets to predict the performance of a classifier on unseen data (without cheating)
  - b. Explain the difference between (1) training error, (2) validation error, (3) cross-validation error, (4) test error, and (5) true error
  - c. For a given learning technique, identify the model, learning algorithm, parameters, and hyperparameters
  - d. Select an appropriate algorithm for optimizing (aka. learning) hyperparameters
- 5. Perceptron
  - a. Explain the difference between online learning and batch learning
  - b. Implement the perceptron algorithm for binary classification [CIML]
  - c. Determine whether the perceptron algorithm will converge based on properties of the dataset, and the limitations of the convergence guarantees
  - d. Describe the inductive bias of perceptron and the limitations of linear models
  - e. Draw the decision boundary of a linear model
  - f. Identify whether a dataset is linearly separable or not
  - g. Defend the use of a bias term in perceptron (shifting points after projection onto weight vector)

## ML as Optimization

- 1. Linear Regression
  - a. Design k-NN Regression and Decision Tree Regression
  - b. Implement learning for Linear Regression using three optimization techniques: (1) closed form, (2) gradient descent, (3) stochastic gradient descent
  - c. Choose a Linear Regression optimization technique that is appropriate for a particular dataset by analyzing the tradeoff of computational complexity vs. convergence speed
  - d. Identify situations where least squares regression has exactly one solution or infinitely many solutions
- 2. Optimization for ML (Linear Regression)
  - a. Apply gradient descent to optimize a function
  - b. Apply stochastic gradient descent (SGD) to optimize a function
  - c. Apply knowledge of zero derivatives to identify a closed-form solution (if one exists) to an optimization problem

- d. Distinguish between convex, concave, and nonconvex functions
- e. Obtain the gradient (and Hessian) of a (twice) differentiable function
- 3. Logistic Regression (Probabilistic Learning)
  - a. Apply the principle of maximum likelihood estimation (MLE) to learn the parameters of a probabilistic model
  - b. Given a discriminative probabilistic model, derive the conditional log-likelihood, its gradient, and the corresponding Bayes Classifier
  - c. Explain the practical reasons why we work with the **log** of the likelihood
  - d. Implement logistic regression for binary or multiclass classification
  - e. Prove that the decision boundary of binary logistic regression is linear
- 4. Feature Engineering / Regularization
  - a. Engineer appropriate features for a new task
  - b. Use feature selection techniques to identify and remove irrelevant features
  - c. Identify when a model is overfitting
  - d. Add a regularizer to an existing objective in order to combat overfitting
  - e. Explain why we should **not** regularize the bias term
  - f. Convert linearly inseparable dataset to a linearly separable dataset in higher dimensions
  - g. Describe feature engineering in common application areas

## Deep Learning

- 1. Neural Networks
  - a. Explain the biological motivations for a neural network
  - b. Combine simpler models (e.g. linear regression, binary logistic regression, multinomial logistic regression) as components to build up feed-forward neural network architectures
  - c. Explain the reasons why a neural network can model nonlinear decision boundaries for classification
  - d. Compare and contrast feature engineering with learning features
  - e. Identify (some of) the options available when designing the architecture of a neural network
  - f. Implement a feed-forward neural network
- 2. Backpropagation
  - a. Differentiate between a neural network diagram and a computation graph
  - b. Construct a computation graph for a function as specified by an algorithm
  - c. Carry out the backpropagation on an arbitrary computation graph
  - d. Construct a computation graph for a neural network, identifying all the given and intermediate quantities that are relevant
  - e. Instantiate the backpropagation algorithm for a neural network
  - f. Instantiate an optimization method (e.g. SGD) and a regularizer (e.g. L2) when the parameters of a model are comprised of several matrices corresponding to different layers of a neural network
  - g. Apply the empirical risk minimization framework to learn a neural network
  - h. Use the finite difference method to evaluate the gradient of a function

- i. Identify when the gradient of a function can be computed at all and when it can be computed efficiently
  - j. Employ basic matrix calculus to compute vector/matrix/tensor derivatives.
3. Deep Learning
- a. Implement the common layers found in Convolutional Neural Networks (CNNs) such as linear layers, convolution layers, max-pooling layers, and rectified linear units (ReLU)
  - b. Explain how the shared parameters of a convolutional layer could learn to detect spatial patterns in an image
  - c. Describe the backpropagation algorithm for a CNN
  - d. Identify the parameter sharing used in a basic recurrent neural network, e.g. an Elman network
  - e. Apply a recurrent neural network to model sequence data
  - f. Differentiate between an RNN and an RNN-LM

## Learning Theory

1. Learning Theory: PAC Learning
- a. Identify the properties of a learning setting and assumptions required to ensure low generalization error
  - b. Distinguish true error, train error, test error
  - c. Define PAC and explain what it means to be approximately correct and what occurs with high probability
  - d. Define sample complexity
  - e. Apply sample complexity bounds to real-world learning examples
  - f. Distinguish between a large sample and a finite sample analysis
  - g. Theoretically justify regularization

## Generative Models

1. Oracles, Sampling, Generative vs. Discriminative
- a. Sample from common probability distributions
  - b. Write a generative story for a generative or discriminative classification or regression model
  - c. Pretend to be a data generating oracle
  - d. Provide a probabilistic interpretation of linear regression
  - e. Use the chain rule of probability to contrast generative vs. discriminative modeling
  - f. Define maximum likelihood estimation (MLE) and maximum conditional likelihood estimation (MCLE)
  - g. For linear regression, show that the parameters which minimize squared error are equivalent to those that maximize conditional likelihood
2. MLE and MAP
- a. Recall probability basics, including but not limited to: discrete and continuous random variables, probability mass functions, probability density functions, events vs. random

- variables, expectation and variance, joint probability distributions, marginal probabilities, conditional probabilities, independence, conditional independence
  - b. Describe common probability distributions such as the Beta, Dirichlet, Multinomial, Categorical, Gaussian, Exponential, etc.
  - c. State the principle of maximum likelihood estimation and explain what it tries to accomplish
  - d. State the principle of maximum a posteriori estimation and explain why we use it
  - e. Derive the MLE or MAP parameters of a simple model in closed form
3. Naive Bayes
- a. Write the generative story for Naive Bayes
  - b. Create a new Naive Bayes classifier using your favorite probability distribution as the event model
  - c. Apply the principle of maximum likelihood estimation (MLE) to learn the parameters of Bernoulli Naive Bayes
  - d. Motivate the need for MAP estimation through the deficiencies of MLE
  - e. Apply the principle of maximum a posteriori (MAP) estimation to learn the parameters of Bernoulli Naive Bayes
  - f. Select a suitable prior for a model parameter
  - g. Describe the tradeoffs of generative vs. discriminative models
  - h. Implement Bernoulli Naive Bayes
  - i. Employ the method of Lagrange multipliers to find the MLE parameters of Multinomial Naive Bayes
  - j. Describe how the variance affects whether a Gaussian Naive Bayes model will have a linear or nonlinear decision boundary

## Graphical Models

1. Hidden Markov Models
- a. Show that structured prediction problems yield high-computation inference problems
  - b. Define the first order Markov assumption
  - c. Draw a Finite State Machine depicting a first order Markov assumption
  - d. Derive the MLE parameters of an HMM
  - e. Define the three key problems for an HMM: evaluation, decoding, and marginal computation
  - f. Derive a dynamic programming algorithm for computing the marginal probabilities of an HMM
  - g. Interpret the forward-backward algorithm as a message passing algorithm
  - h. Implement supervised learning for an HMM
  - i. Implement the forward-backward algorithm for an HMM
  - j. Implement the Viterbi algorithm for an HMM
  - k. Implement a minimum Bayes risk decoder with Hamming loss for an HMM
2. Bayesian Networks
- a. Identify the conditional independence assumptions given by a generative story or a specification of a joint distribution
  - b. Draw a Bayesian network given a set of conditional independence assumptions

- c. Define the joint distribution specified by a Bayesian network
- d. Use domain knowledge to construct a (simple) Bayesian network for a real-world modeling problem
- e. Depict familiar models as Bayesian networks
- f. Use d-separation to prove the existence of conditional independencies in a Bayesian network
- g. Employ a Markov blanket to identify conditional independence assumptions of a graphical model
- h. Develop a supervised learning algorithm for a Bayesian network
- i. Use samples from a joint distribution to compute marginal probabilities
- j. Sample from the joint distribution specified by a generative story
- k. Implement a Gibbs sampler for a Bayesian network

## Reinforcement Learning

1. Reinforcement Learning: Value & Policy Iteration
  - a. Compare the reinforcement learning paradigm to other learning paradigms
  - b. Cast a real-world problem as a Markov Decision Process
  - c. Depict the exploration vs. exploitation tradeoff via MDP examples
  - d. Explain how to solve a system of equations using fixed point iteration
  - e. Define the Bellman Equations
  - f. Show how to compute the optimal policy in terms of the optimal value function
  - g. Explain the relationship between a value function mapping states to expected rewards and a value function mapping state-action pairs to expected rewards
  - h. Implement value iteration
  - i. Implement policy iteration
  - j. Contrast the computational complexity and empirical convergence of value iteration vs. policy iteration
  - k. Identify the conditions under which the value iteration algorithm will converge to the true value function
  - l. Describe properties of the policy iteration algorithm
2. Reinforcement Learning: Q-Learning
  - a. Apply Q-Learning to a real-world environment
  - b. Implement Q-learning
  - c. Identify the conditions under which the Q-learning algorithm will converge to the true value function
  - d. Adapt Q-learning to Deep Q-learning by employing a neural network approximation to the Q function
  - e. Describe the connection between Deep Q-Learning and regression

## Learning Paradigms

1. PCA and Dimensionality Reduction

- a. Define the sample mean, sample variance, and sample covariance of a vector-valued dataset
  - b. Identify examples of high dimensional data and common use cases for dimensionality reduction
  - c. Draw the principal components of a given toy dataset
  - d. Establish the equivalence of minimization of reconstruction error with maximization of variance
  - e. Given a set of principal components, project from high to low dimensional space and do the reverse to produce a reconstruction
  - f. Explain the connection between PCA, eigenvectors, eigenvalues, and covariance matrix
  - g. Use common methods in linear algebra to obtain the principal components
2. K-Means
- a. Distinguish between coordinate descent and block coordinate descent
  - b. Define an objective function that gives rise to a "good" clustering
  - c. Apply block coordinate descent to an objective function preferring each point to be close to its nearest objective function to obtain the K-Means algorithm
  - d. Implement the K-Means algorithm
  - e. Connect the nonconvexity of the K-Means objective function with the (possibly) poor performance of random initialization
3. Ensemble Methods: Bagging
- a. Distinguish between (sample) bagging, the random subspace method, and random forests.
  - b. Implement (sample) bagging for an arbitrary base classifier/regressor.
  - c. Implement the random subspace method for an arbitrary base classifier/ regressor.
  - d. Implement random forests.
  - e. Contrast out-of-bag error with cross-validation error.
  - f. Differentiate boosting from bagging.
  - g. Compare and contrast weighted and unweighted majority vote of a collection of classifiers.
  - h. Discuss the relation in bagging between the sample size and variance of the base classifier/regressor.
  - i. Bound the generalization error of a random forest classifier.
4. Ensemble Methods: Boosting
- a. Explain how a weighted majority vote over linear classifiers can lead to a non-linear decision boundary
  - b. Implement AdaBoost
  - c. Describe a surprisingly common empirical result regarding Adaboost train/test curves
5. Recommender Systems
- a. Compare and contrast the properties of various families of recommender system algorithms: content filtering, collaborative filtering, neighborhood methods, latent factor methods
  - b. Formulate a squared error objective function for the matrix factorization problem
  - c. Implement unconstrained matrix factorization with a variety of different optimization techniques: gradient descent, stochastic gradient descent, alternating least squares

- d. Offer intuitions for why the parameters learned by matrix factorization can be understood as user factors and item factors

## References

Several of these learning objectives are copied or adapted from [Daume III \(2018\) "CIML"](#).