Introduction to Machine Learning

CMSC 422

MARINE CARPUAT
marine@cs.umd.edu

End of semester logistics

Final Exam

- Wednesday May 17th, 10:30am – 12:30pm, CSIC 3117
- closed book, 1 double-sided page of notes
- cumulative, with a focus on topics covered in 2nd half of semester
  - linear models, gradient descent
  - probabilistic models
  - unsupervised learning (PCA)
  - neural networks
  - kernels, SVMs
  - bias and adaptation
  - deep learning

End of semester logistics

- Course evals

https://www.CourseEvalUM.umd.edu

- See piazza for practice problems

What you should know: Linear Models

- What are linear models?
  - a general framework for binary classification
  - how optimization objectives are defined
    - loss functions and regularizers
  - separate model definition from training algorithm (Gradient Descent)

What you should know: Gradient Descent

- Gradient descent
  - a generic algorithm to minimize objective functions
  - what are the properties of the objectives for which it works well?
  - subgradient descent (ie what to do at points where derivative is not defined)
  - why choice of step size, initialization matter

What you should know: Probabilistic Models

- The Naïve Bayes classifier
  - Conditional independence assumption
  - How to train it?
  - How to make predictions?
  - How does it relate to other classifiers we know?

- Fundamental Machine Learning concepts
  - iid assumption
  - Bayes optimal classifier
  - Maximum Likelihood estimation
  - Generative story

What you should know: Neural Networks

- What are Neural Networks?
  - Multilayer perceptron
- How to make a prediction given an input?
  - Forward propagation: Matrix operations + non-linearities
- Why are neural networks powerful?
  - Universal function approximators!
- How to train neural networks?
  - The backpropagation algorithm
    - How to step through it, and how to derive update rules

What you should know: PCA

- Principal Components Analysis
  - Goal: Find a **projection** of the data onto directions that **maximize variance** of the original data set
  - PCA **optimization objectives** and resulting **algorithm**
  - Why this is useful!

What you should know: Kernels

- Kernel functions
  - What they are, why they are useful, how they relate to feature combination

- Kernelized perceptron
  - You should be able to derive it and implement it

What you should know: SVMs

- What are Support Vector Machines
  – Hard margin vs. soft margin SVMs
- How to train SVMs
  – Which optimization problem we need to solve
- Geometric interpretation
  - What are support vectors and what is their relation with parameters **w**,b?
- How do SVM relate to the general formulation of linear classifiers
- Why/how can SVMs be kernelized

Example questions for understanding SVMs

- After training a SVM, we can discard all examples which are not support vectors and can still classify new examples. True or False?

- When the training data is not completely linearly separable, what happens if we train a hard SVM (i.e. the SVM without slack variables)?

- Consider the primal non-linearly separable version of the SVM objective. What do we need to do to guarantee that the resulting model is linearly separable?

What you should know: Bias and how to deal with it

- What is the impact of data selection bias on machine learning systems?
- How to address train/test mismatch
  - Unsupervised adaptation
    - Using auxiliary classifier
  - Supervised adaptation
    - Feature augmentation

What you should know: Deep Learning

- Neural network architectures can encode inductive bias relevant to specific tasks (e.g., vision, language), and enable end-to-end training
  - Convolutional Neural Networks
  - Recurrent Neural Networks

- Why training deep networks is challenging
  - Computationally expensive, vanishing gradient

- Stochastic gradient descent

Machine Learning

- Paradigm: “Programming by example”
  - Replace ``human writing code'' with ``human supplying data''
- Most central issue: generalization
  - How to abstract from ``training'' examples to ``test'' examples?

Course Goals

- By the end of the semester, you should be able to
  - Look at a problem
  - Identify if ML is an appropriate solution
  - If so, identify what types of algorithms might be applicable
  - Apply those algorithms
- This course is not
  - A survey of ML algorithms
  - A tutorial on ML toolkits such as Weka, TensorFlow, …

Key ingredients needed for learning

- Training vs. test examples
  - Memorizing the training examples is not enough!
  - Need to generalize to make good predictions on test examples
- Inductive bias
  - Many classifier hypotheses are plausible
  - Need assumptions about the nature of the relation between examples and classes

Machine Learning as Function Approximation

Problem setting

- Set of possible instances X
- Unknown target function $f: X \rightarrow Y$
- Set of function hypotheses $H = \{h \mid h: X \rightarrow Y\}$

Input

- Training examples $\{(x^{(1)}, y^{(1)}), \ldots (x^{(N)}, y^{(N)})\}$ of unknown target function f

Output

- Hypothesis $h \in H$ that best approximates target function f

Formalizing Induction

- Given
  - a loss function l
  - a sample from some unknown data distribution D

- Our task is to compute a function f that has low expected error over D with respect to l.

$$\mathbb{E}_{(x,y)\sim D}\{l(y,f(x))\} = \sum_{(x,y)} D(x,y)l(y,f(x))$$

Beyond 422…

- Many relevant courses in machine learning and applied machine learning in CS@UMD
  - Artificial Intelligence (CMSC 421), Robotics (CMSC498F), Language (CMSC289J , CMSC 723), Vision (CMSC 426), …
- Experiment with tools and datasets
  - weka, scikit-learn, vowpal wabbit, theano, pyTorch, tensorflow…
  - kaggle…
- Keep up to date on cutting-edge machine learning
  - Attend research seminars in the department (e.g., go.umd.edu/cliptalks)
  - Talking Machines podcast

Beyond 422…

- Machine learning is everywhere
- Many opportunities to create new high impact applications

- But challenging issues arise
  - Fairness
  - Accountability
  - Transparency
  - Privacy