https://introml.mit.edu/

6.390 Intro to Machine Learning

Final Review

Shen Shen

May 14, 2024

Course Evaluations

http://registrar.mit.edu/subjectevaluation

We'd love to hear your thoughts on the course: this provides valuable feedback for us and other students, for future semesters! Thank you!🙏

Outline

- Rundown
- Past Exam
- Q&A

Week 1 - IntroML

- Terminologies
  - Training, validation, testing
  - Identifying overfitting, underfitting
- Concrete process
  - Learning algorithm
  - Cross-validation
  - Concept of hyperparamter

Week 2 - Regression

- Problem Setup
- Analytical solution formula $\theta^* = \left(\tilde{X}^\top \tilde{X}\right)^{-1} \tilde{X}^\top \tilde{Y}$ (what's $\tilde{X}$)
- When $\tilde{X}^\top \tilde{X}$ not invertible (solutions still exist; just not via the "formula")
  - Practically (two scenarios)
  - Visually (obj fun no longer "bowl" shape, "half-pipe" shape)
  - Mathematically (loss of solution uniqueness)
- Regularization
  - Motivation, how to, when to.
- Cross-validation

Week 3 - Gradient Descent

- Gradient vector
- The algorithm, gradient-descent formula
- How does "stochastic" gradient descent differ
- (Convex + small-enough step-size + gradient descent) guarantees convergence to global min (when global min exists)
  - If not convex, can e.g. get stuck in local min
  - If step-size too big, can diverge
  - If stochastic gradient descent, can be "wild"

Week 4 - Classification

- (Binary) linear classifier (sign based)
- (Binary) Linear logistic classifier
    - Sigmoid
    - NLL loss
- Linear separator (equation form, pictorial form with normal vector)
- Linear separability
- How to handle multiple classes
    - Softmax generalization
    - Multiple sigmoids
    - One-vs-one, one-vs-all

Week 5 - Features

- Feature transformations
  - Applying a fixed feature transformation
  - Hand-design a good feature transformation (e.g. towards getting linear separability)
  - Interplay between number of features, quality of features, and quality of learning algorithms
- Feature encoding
  - One-hot, thermometer, factored, numerical, standardization

Week 6 - Neural Networks

- Forward-pass (for evaluation)
- Backward-pass (via backpropogation, for optimization)
- Source of expressiveness
- Output layer design
  - dimension, activation, loss
- Hand-design weights
  - to match some given function form
  - achieve some goal (e.g. separate a given data set)

Week 7 - CNN

- The convolution operation
  - various hyper-parameters (filter size, padding size, stride) in spatial dimension
  - the 3rd channel/depth dimension
  - reason about in/out shapes.
- Tailored for vision problems: fully-connected nets + weight sharing.
- Convolutional filters: "Pattern matching" template
- Forward pass: convolution operation; backward: backprop to learn filter weights/bias as usual.
- Independent and parallel processing.
- Max-pooling and typical "pyramid" stack.

Week 8 - Transformers

- parallel-processing machines
- a single input (sentence, image) is tokenized into a sequence: n tokens, each token in d dimensional
- attention mechanism
  - learn three weights W_q, W_k, W_v to turn raw inputs into query, key, value
  - the mechanics; shapes, softmax, number-crunching
  - the idea of masking

Week 9 - Non-parametric methods

- Decision trees:
  - Flow chart; if/else statement; human-understandable
  - Split dimension, split value, tree structure (root/decision node and leaf)
  - For classification, weighted-average-entropy/accuracy; for regression, MSE.
- $k-$nearest neighbors:
  - memorizes data
  - inefficient in test/prediction time

Week 10 - MDPs

-
  - Definition (the five tuple)
  - π, V, and Q : definition and interpretation
  - Policy evaluation: given $\pi(s)$, calculate $V(s)$
    - via summation, or
    - via Bellman recession (for finite-horizon) or equation (for infinite horizon)
  - Policy optimization: finding optimal policy $\pi^*(s)$
    - Toy setup: solve via heuristics
    - More generally: Q value-iteration
  - Interpretation of optimal policy:
    - how various setup changes optimal policy R, γ, h

Week 11 - Reinforcement Learning

- How RL setup differs from MDP
- Q-learning algorithm
  - Forward thinking: given experiences, work out Q-values.
  - Backward thinking: given realized Q-values, work out experiences.
  - Two new hyper-parameters (compared with MDP):
    - $\epsilon-$greedy action selection
    - α the learning rate
- The idea of fitting parameterized Q-functions via regression
  - can handle larger/continuous state/action space

Week 12 - Unsupervised Learning

- Unsupervised learning setup
- Clustering:
  - The clustering algorithm (cluster assignment; cluster center updates)
  - Initialization matters
  - The choice of hyper-parameter k matters
- Auto-encoder:
  - The idea of compression->reconstruction
  - Mechanically, exactly the same as any vanilla neural architecture.

https://shenshen.mit.edu/tree

(The demo won't embed in PDF. But the direct link below works.)

https://shenshen.mit.edu/tree

Resources

- All the released materials Week 1 - Week 12
- Review Question Sampler

```python
import random
terms= ["fall2023", "spring2023", "fall2022", "spring2022",
        "fall2021", "fall2019", "fall2018", "fall2018"]

qunums = range(1,9)
base_URL = "https://introml.mit.edu/_static/spring24/final/review/final-"

term = random.choice(terms)
num = random.choice(qunums)
print("term:", term)
print("question number:", num)
print(f"Link: {base_URL+term}.pdf")
```

- Past exams
- Annotated Lecture Notes Study Questions
- Annotated Lab takeaways
- Piazza and OHs

General problem-solving tips

Polya's Problem Solving Techniques

In 1945 George Polya published the book *How To Solve It* which quickly became his most prized publication. It sold over one million copies and has been translated into 17 languages. In this book he identifies four basic principles of problem solving.

Polya's First Principle: Understand the problem

This seems so obvious that it is often not even mentioned, yet studens are often stymied in their efforts to solve problems simply because they don't understand it fully, or even in part. Polya taught teachers to ask students questions such as:

- Do you understand all the words used in stating the problem?
- What are you asked to find or show?
- Can you restate the problem in your own words?
- Can you think of a picture or diagram that might help you understand the problem?
- Is there enough information to enable you to find a solution?

Polya's Second Principle: Devise a plan

Polya mentions that there are many reasonable ways to solve problems. The skill at choosing an appropriate strategy is best learned by solving many problems. You will find choosing a strategy increasingly easy. A partial list of strategies is included:

- Guess and check
- Make an orderly list
- Eliminate possibilities
- Use symmetry
- Consider special cases
- Use direct reasoning
- Solve an equation
- Look for a pattern
- Draw a picture
- Solve a simpler problem
- Use a model
- Work backwards
- Use a formula
- Be ingenious

Polya's Third Principle: Carry out the plan

This step is usually easier than devising the plan. In general, all you need is care and patience, given that you have the necessary skills. Persist with the plan that you have chosen. If it continues not to work discard it and choose another. Don't be misled, this is how mathematics is done, even by professionals.

Polya's Fourth Principle: Look back

Polya mentions that much can be gained by taking the time to reflect and look back at what you have done, what worked, and what didn't. Doing this will enable you to predict what strategy to use to solve future problems.

More detailed CliffsNotes

General exam tips

- Arrive 5min early to get settled in.
- Bring a watch.
- Bring a pencil (and an **eraser**).
- Look over whole exam and strategize for the order you do problems.
- Bring some water.

6.390

https://introml.mit.edu/

Thanks for the semester.

🍀 Best of luck!