Getting started with Machine Learning

CIS400/600, Jan. 25-27,2022

Overview

- What it’s all about
- Models
- Algorithms
    - Architecture
    - Parameters
- Applications
- Evolutionary Machine Learning

What it’s all about: ML is Data-Driven!

- Knowledge required to solve (new) problems must come from data.
- In some applications, experts can be queried (knowledge elicitation).
- We often begin with a preconceived notion of what the knowledge model must look like.
- If no rough notion is available, we hypothesize, based on other problems that we have solved before.
- Fine-tuning requires looking at available data.
- If the result is unsatisfactory, we revise our model, sometimes drastically.
- Machine learning must accommodate for these processes!

Reality

Processes generate data!

- Some processes are deterministic; most are not.
- Some processes involve the actions of autonomous agents.
- Noise may be inherent or the result of measurement.
- The nature of a process may change with time.

Models

- Models approximate reality using equations, figures, diagrams, etc.
- Models can be tested against data.
- Evaluation criteria depend on the problem.
- Examples of models:
  - Differential equations
  - Production rules
  - Hidden Markov models
  - Feedforward Neural Networks
  - Associative networks
  - Decision trees

Learning Algorithms

- Learning algorithms use data to determine characteristics of the model architecture, and to modify model parameters.

- Architecture learning examples:
  - How many layers and nodes in a feedforward neural network?
  - How many rules, and of what kind, in a production system?
- Parameter learning examples:
  - Coefficients in an equation
  - Weights in a neural network

Class Exercise

Learning task: to predict how much snow is expected tomorrow.

- Questions:
  1. What data can be used?
  2. What kind of model?
  3. What details of the model architecture need to be determined?
  4. What are the parameters to be learnt?

What kind of data?

- Unsupervised, no labeled data
- Supervised: when substantial amount of labeled data is available
- Semi-supervised: small amount of labeled data, much more unlabeled data
- Reinforcement-driven: the results of an action can be evaluated

Class Exercise

Learning task: to predict how much snow is expected tomorrow.

- What data can we use?
- Is this supervised/unsupervised/semi-supervised/reinforcement learning?

Criteria for evaluation

- Mean squared error (and other norms)
- Classification accuracy (possibly weighted)
- Confusion matrix
- Distances between points clustered together
- Information theoretic (e.g., cross-entropy, Kullback-Leibler divergence)

- Training vs. testing vs. cross-validation

- Multi-objective optimization (weighted sum vs. domination)

Class Exercise

Learning task: to predict how much snow is expected tomorrow.

Assume we have somehow accomplished learning.

- How would we evaluate the result of learning?

Learning Approaches

- Point-based (e.g., gradient descent, hill climbing, …)
- Ensemble (combining results from multiple models trained separately)
- Population-based (e.g., evolutionary)

Need to choose models and learning algorithms

- The same problem may be addressed using different models!

- The same model’s parameter values can be estimated using different learning algorithms.

Domain-specific information is important!

For example, consider three short-term forecasting problems.

- **Finance:** a company’s stock price is likely to be about the same tomorrow as today. (Why?)

- **Clothing:** I am not likely to wear the same shirt on Thursday as Tuesday! (Why?)

- **Weather:** If it snows today, will it snow tomorrow?

No Free Lunch Theorems [Wolpert and Macready, 1996-7]

Paraphrased: *When all possible functions are equally likely, all search algorithms (attempting to find the desired function from these) are equally good.*

***Proof:* For every example where algorithm 1 does better than algorithm 2, we can construct a counter-example where 1 is worse.**

Caveat: this may not be the case if the algorithms are adaptive, or if we restrict ourselves to examples that arise from natural or reasonable processes (with nice properties such as continuity).

In practice: assume that an algorithm that worked well for one problem will also work well for other similar problems.

Model selection criteria

- Adequacy and expressive power (what it is capable of representing);
- Number of trainable parameters;
- Information complexity: number of bits needed to describe the model;
- How to find the right settings for hyper-parameter values;
- Robustness: what happens if a part of the model “fails”, or the input data is noisy, or the environment changes;
- Quality of solutions obtained;
- Computational effort required to obtain solutions;
- Ability to represent problem-specific knowledge; and
- Ability to build a hardware implementation.

On model complexity

- Occam's razor: prefer simpler models, with fewer parameters
  - Avoid over-fitting
  - Expected to perform better on test data

- But: what if the more complex model gives better quality results, and obtains them faster?

- We must consider the tradeoff!

Amount of data vs. number of parameters

- Rule-of-thumb: amount of data needs to be at least ten times the number of trainable parameters.

- Theoretical results (for neural networks): …exponential…

Strategies for finding the right size model

- Start small, and grow (iteratively).
- Start big, and prune (iteratively).
- Regularize: add a size penalty term to the loss function (or other quantity being optimize).

Gradient Descent

If minimizing a loss function, parameters Θ are modified in a direction opposite to the first derivative (gradient)

$$\Theta_{t+1} = \Theta_t - c.\ d(loss)/d\Theta_t$$

The result decreases the loss by a small amount.

This process is repeated, each time recalculating the gradient.

The error back-propagation algorithm used for training feedforward neural networks implements this systematically, beginning with the layer of weights closest to the outputs, and working backwards towards the inputs.

Greedy or best-first or hill-climbing algorithms

- If the problem is not continuous, a gradient may not exist, but we may still pretend that it does, assuming linear behavior locally at each point.
- Repeat: examine all “neighbors” of the current state, choose the best.

Well-known problem…

- If a function being optimized has multiple local optima, gradient descent (and hill-climbing, etc.) can only find the nearest one, and search stops well before reaching the best possible solution (global optimum).

- This motivates other “global search” algorithms such as evolutionary algorithms and simulated annealing.

Class Exercise

- Mean squared error is a quadratic function.
- Every quadratic function has a unique optimum, not multiple local optima.
- Therefore gradient descent will always succeed in finding the global optimum (e.g., with neural network training).

Comment?

Typical problems we address using (evolutionary) machine learning—examples from Finance

- Classification (e.g., recognizing whether to give credit to a loan applicant)
- Function approximation (e.g., deciding what interest rate to charge a loan applicant with a known risk profile)
- Forecasting (e.g., predicting next year's interest rates based on previous years' interest rates and other financial/ fiscal/ political/ economic information)
- Optimization (e.g., determining how much of different mutual funds to purchase in order to maximize returns based on a customer's risk tolerance)

Evaluating 2-class classification performance in terms of True/False Positives/Negatives

- *Confusion matrix: [[TP FN][FP TN]]*
- *Sensitivity* (or Recall) = $TP/(TP + FN)$,
- *Specificity* = $TN/(TN + FP)$, and
- *Precision* = $TP/(TP + FP)$.
- Accuracy = $(TP + TN)/(TP + FP + TN + FN)$,
- *F1 Score* = $(2TP)/(2TP + FN + FP)$, and
- *Positive Likelihood Ratio* = $TP(FP + TN)/FP(TP + FN)$.

[Note: “Positive” often means “bad” in medical/financial problems!]

Class Exercise

- What about multi-class problems?
- Accuracy=?

Class Exercise

Assume 90 “non-fraudulent” (negative), and 10 “fraudulent” (positive) transactions.

Which of the following algorithms is “better”?

- Algorithm 1: TN=90, FN=10, TP=0, FP=0, Accuracy=90%
- Algorithm 2: TN=70, FN=0, TP=10, FP=20, Accuracy=80%

- CONCLUSION?

Class Exercise

Assume 50 patients with Covid (Positive class), and 50 without Covid (Negative class).

Which of the following Covid tests is “better”?

- **Test 1**: TP=50, FP=10, TN=40, FN=0, Accuracy=90%
- **Test 2:** TP=45, FP=5, TN=48, FN=2, Accuracy=93%

- CONCLUSION?

My conclusion

- Instead of just Accuracy, we need to use other measures appropriate for the problem.

- Specific strategies must be used to increase the emphasis given to the class with fewer points, or for which errors are more worrisome, e.g.,
  - Modifying the loss function, e.g., with asymmetric penalties
  - Over-sampling the smaller (or more important) class
  - Under-sampling the larger (or less important) class—but we may lose useful information when we ignore some data

MSE vs. Accuracy for classification problems

- Assume that we’d like to maximize accuracy for a classification problem (e.g., character recognition, with equal number of examples for each character).

- Does minimizing MSE guarantee maximizing accuracy?

- What are the implications for using backpropagation (or other gradient descent procedures) to address classification problems?

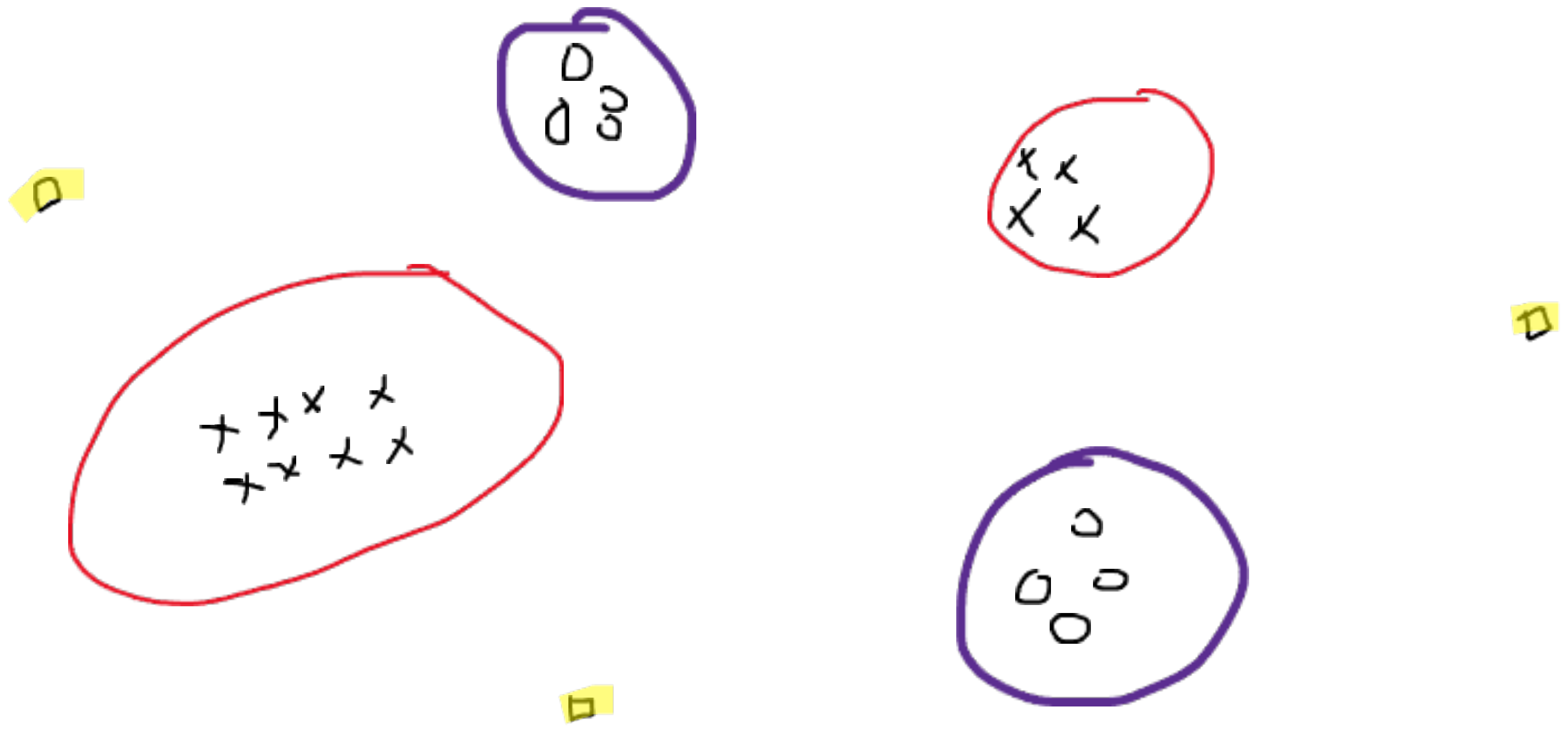
Other concerns for classification problems

- The output of a neural network is not a probability! Calibration may be possible, e.g., “what fraction of data belong to class 1 when an NN output is in the range 0.7-0.8”?
- Noise: training data may be corrupted by noise or errors.
- Bias: experts have subjective opinions that bias the data they provide; training data may reflect bad choices made by humans.
- Corruption of data: normalization or other deliberate changes to data may result in loss of information needed for learning.
- Extrapolation is notoriously more difficult than interpolation.

Anomaly detection vs. classification

- In some problems, one “normal” class is well defined, while the other “abnormal” class is only defined in terms of data points being very different from those of the normal class.
- Within the abnormal class, there may be drastic differences between points!
- For such problems, choosing a classic feedforward neural network model would not help.
- Instead, modeling the normal class using clusters would be better, followed by estimating for each point the probability that it belongs to the normal class’s clusters.

Classification with clusters



When clustering is used...

- We must balance between three criteria:
  - maximizing inter-cluster distance,
  - minimizing intra-cluster distance,
  - minimizing the number of clusters.
- Inter-cluster distance may be defined in different ways (e.g., distance between centroids, or distance between cluster boundaries) and may also have to account for the incommensurability of different dimensions (e.g., human height in meters, human weight in kilograms).
- Intra-cluster distance may focus on average or maximum distances between points in a cluster.

If you are evaluating a new algorithm or model, what is the “straw man” for comparison?

- Classification: linear perceptron; support vector machine; small feedforward network trained with backpropagation; decision tree; random forest
- Function approximation: linear regression; logistic regression; support vector regression; small feedforward NN (with backpropagation)
- Forecasting/prediction over time: no change model; short-term average; long-term average; seasonal average; linear trend model; ARMA; ARIMA; LSTM/GRU neural network

How to show results

- Most algorithms show fast improvements in early iterations and slow or sporadic improvements later. Logarithmic plots are better than linear plots.
- Visual comparisons across two separate graphs are hard. Show multiple plots on the same graph, using different colors, dashed/dotted lines, etc.
- The reader may not know what conclusion is to be drawn from a graph or table. State the conclusion clearly.
- Tables with many numbers are hard to read. Avoid using too many digits; approximate instead (e.g., “12.1k” instead of “12,127.8149”).
- Each trial with an algorithm may yield different results. Hence average across many trials (preferably 30), and also show standard deviations.
- Sometimes extreme values skew the averages. The median is then more informative than the mean.

Tradeoffs in Machine Learning

- Stability-Plasticity Dilemma: keep the old vs. learn the new?
- Bias-Variance Dilemma: how accurate should results on training data be?
- Quality of results vs. Computational effort

Evolutionary Machine Learning

- An optimization problem is formulated.
- Multiple candidate solutions (***individuals***) are considered simultaneously.
- Each individual is evaluated (***fitness***)
- Different individuals may interact.
- Information may be gleaned from a group of individuals.
- Additional attributes of an individual may be used (memory, velocity)
- Changes are made in individuals over multiple iterations (*generations)*.
- *Selection pressure* drives the *population* towards better individuals.

Terminological quirks

- *Fitness* to be maximized vs. *cost* to be minimized—unfortunately, people often use the word “fitness” along with minimization goals.
- *Chromosomes* vs. *individuals* vs. *particles*
- *Genetic* vs. *Evolutionary algorithms*
- *Swarm* vs. *population* vs. *sample*
- *Evolution vs. adaptation*
- Adaptation of *Hyper-parameters*