ML Review

Machine Learning for Trading (CS 7641)

- example x's: price, bollinger, momentum; y's: return and future price
- Supervised regression learning: linear regression, KNN, trees, forests
- RL
  - Markov decision problem: set of states (in market), set of actions (buy, sell, nothing), transition function T(state, action, state), reward function R(state, action)
  - Q Learning: Q'[s,a] = (1-alpha) * Q[s,a] + alpha * improved_estimate
- Induction, deduction, supervised learning, unsupervised learning (news, genes), RL, decision trees
- Definitions: instances (input), concept (function), target concept (answer), hypothesis (set of all possible concepts), sample (training set), candidate (possible target concept), testing set. Tree nodes = attributes, edges = values
- Preprocessing (clean, train/test) -> Learning (model, cross-validate, performance, params optimization) -> Evaluation (against test) -> Prediction
- Target function: type of knowledge to be learned (e.g. each possible checkers board score)
- Representation for target function (e.g. linear function of board artifacts, # of pieces, # of pieces threatened)
- Learning mechanism: gradient descent
- Decision Tree: Robust to errors and missing attributes
- Entropy: sum(-p*logp)
- Overfit -> Occam's razor
- NN: Well suited for noisy, complex sensor data, such as inputs from cameras and microphones
- Gradient descent derivation
- Sigmoid: 1/(1-e^-y)
- Instance based learning (KNN): training data in storage and use it to make a forecast
- Other algorithms reduce data into a function, then make predictions ignoring the data
- Bayesian Learning: P(h | D) = P(D | h) * P(h) / P(D)

$$P(disease|pos) = 21\% = 98\% * 0.8\% / (98\% * 0.8\% + 3\% * 99.2\%)$$

- Pros/cons: DT, KNN, Boosting, SVM, NN, Bayes, Random Opt/Annealing
- Review
  - Maximum a posteriori MAP = argmaxh P(D | h) * P(h)
  - Maximum likelihood ML = argmaxh P(D | h)
- EM: Will not diverge, may not converge, can get stuck - must random restart
- K-means: randomly select k centers (points), assign each point to cluster (E), recalculate centers (M)
- Feature selection: exponential problem 2^n

- Filtering: selecting features first, then running algo, ignores learning problem, fast. Can use decision tree to select which features give most info. gain, then pass those features to another learner such NN
- Wrapping (optimizing algorithm based on given features and reiterating, slower). forward search (run learner with each feature independently, keep the best, add one more feature in addition to first and iterate as long as error improves)
- Feature transformation: PCA (Maximizes variance, Mutually orthogonal), ICA (Cocktail party problem: extract voice from multiple sources in a party)

Machine Learning (CS 7641)

- PCA: Obtain the Eigenvectors and Eigenvalues from the covariance matrix or correlation matrix. Sort eigenvalues in decreasing order
- Value Iteration, Policy Iteration, and Q-Learning
- Feature selection => curse of dimensionality (amount of data needed is 2^n, where n is the number of features)

Reinforcement Learning (CS 7642)

- RL: model is unknown or difficult to work with directly
- states (S), actions (A), rewards (R)
- methods for solving finite Markov decision problems:
    - dynamic programming: methods are well developed mathematically, but require a complete and accurate model of the environment
    - Monte Carlo methods: methods don't require a model and are conceptually simple, but are not well suited for step-by-step incremental computation
    - temporal-difference learning: methods require no model and are fully incremental, but are more complex to analyze. Monte Carlo methods must wait until the end of the episode to determine the increment, TD methods need wait only until the next time step
- Q-learning. ε-greedy guarantees convergence if

$$\sum \alpha = \infty; \sum \alpha^2 < \infty$$

- Sarsa update

$$Q_t = Q_t + \alpha(r + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

- partially observable Markov decision processes: the agent does not simply know its state and must take actions to gain state-related information

- Q-Learning (off-policy greedy)

- SARSA (on-policy)

- Deep Q Network

- Plans: fixed, conditional (if statements), stationary policy/universal plan (mapping from state -> action, RL focus)

- Summarize policy

  - One sequence: sum of rewards
  - Over many sequences: average expectation

- Evaluate learner: total value, computational complexity (time), sample complexity (time)

- Incremental learning

$$V_T(S_1) = V_{T-1}(S_1) + \alpha_T[R_T(S_1) - V_{T-1}(S_1)]$$

- Where

$$\alpha_T = 1/T$$

- Must satisfy

$$\sum \alpha = \infty; \sum \alpha^2 < \infty$$

- Examples

$$\sum \alpha = 1/T$$

$$\sum \alpha = (1/T)^{2/3}$$

TODO Next notes_2017.pdf p 157

AI (CS 6601)

- AI (pass Turing test): natural language processing, knowledge representation, automated reasoning, machine learning (extrapolate), computer vision, robotics (manipulate objects)
- Minimax time: $O(b^m)$ where b is the number of actions and m number of steps (depth of the tree)
- Search: BFS, DFS, Cheapest first search (explored shortest total length), A*
-
  - Annealing: use temperature as a proxy for randomness. Start with high temperature (high randomness) and slowly lower it (low randomness)
- Travelling salesman: randomly connect cities, then remove any crossings of paths (within 1% of optimal)
- Genetic algorithms: determine score for each state. Higher scores have higher probability to appear in next generations. Crossover with other state for breed new, more optimal states
- Decision theory = probability theory + utility theory
- Bayes rule: P(cause | effect) = P(effect | cause)P(cause) / P(effect)

- ML: KNN, Gaussian, Naive Byers
    - DT: decide each variable to split on based on information gain. Info gain is the expected reduction in entropy
    - Boosting: use weak classifiers and combine then by weighting errors more
    - NN: perceptron and backprop
    - K-means: assigning to clusters and updating means of clusters correspond respectively to the E (expectation) and (maximization) steps of the EM algorithm. O(k*n) where k is the k means and n is number of data
- Learning from Examples
- Cross validation methodology: k-fold training set and test set
- Regularization: process of explicitly penalizing complex hypotheses
- Hidden Markov Models

Data Visualization (CSE 6242)

- 1D data: histogram, strip plot, box plot
- 2D data: scatter plot, line plot
- 3D+: multivariate scatter, facets, contour
- Strip plot: x-axis description, y-axis value
- Histograms: x-axis bar label, y-axis count
- Multivariable scatter: use point type to distinguish 3rd dimension
- Missing Completely At Random (MCAR): probability that data is missing does not depend on observed or unobserved measurements. All solutions possible
- Missing At Random (MAR): probability that data is missing does not depend on unobserved data. May introduce bias
- Solutions:
    - Remove entire row
    - Replace with most likely value
    - Imputation: Build a probability model and sample from the model
- Outliers detection:
    - Below and above certain percentile (robust)
    - More than x standard deviations (requires the computation of mean and median which is not robust given outliers)
- Solutions:
    - Truncate: remove all outliers
    - Winsorization: shrink to the border of the main part of data
    - Robust procedure: keep all data, choose robust procedure (such as median)
- Transformation: Highly skewed distribution -> transformation -> map data to common distribution (Gaussian or Gamma) -> fit model
- power transform: $\frac{x^\lambda - 1}{\lambda}$
    - lambda > 1: convex, removes left skew
    - lambda < 1: concave, removes right skew

    - lambda choice: histogram or log-log plot
- Other transforms: binnning, binarization (0-1), tall vs wide data (easier to analyze, hard to remove)
- Logistic regression
- If data not linearly separable in original dim (eg 2D): increase dimension (x1, x2) => (1, x1, x2, x1^2, x2^2, x1x2)

ML Crash Course Google

- Terminology: labels, features, examples, models
- A regression model predicts continuous values
- A classification model predicts discrete values
- Regression:
    - loss is a number indicating how bad the model's prediction was
    - Mean square error (MSE) is the average squared loss per example over the whole dataset
- Gradient descent, convergence, training, loss
- A Machine Learning model is trained by starting with an initial guess for the weights and bias and iteratively adjusting those guesses until learning the weights and bias with the lowest possible loss.
- gradient always points in the direction of steepest increase in the loss function
- gradient descent algorithm takes a step in the direction of the negative gradient
- Stochastic gradient descent (SGD) uses only a single example (a batch size of 1 per iteration
- TensorFlow: Tensors are N-dimensional (where N could be very large) data structures, most commonly scalars, vectors, or matrices
- Estimator (tf.estimator) High-level, OOP API.
- tf.layers/tf.losses/tf.metrics Libraries for common model components.
- TensorFlow Lower-level APIs
- TensorFlow consists of the following two components:
    - a graph protocol buffer
    - a runtime that executes the (distributed) graph
- These two components are analogous to Python code and the Python interpreter

CONSOLIDATED:

Supervised Learning

Linear Regression

- Variance
    - sample: $\frac{\sum(y_i - \bar{y})^2}{n-2}$

    - costs us 2 degrees of freedom to estimate b0 and b1: therefore, n-2
- R^2
    - quantifies the strength of a linear relationship, R^2 could be 0 but there could be a non-linear relationship
    - value should not be interpreted as meaning that the estimated regression line fits the data well (it might be slightly curved)
    - coefficient of determination R^2 could be greatly affected by just one data point

Source:

- onlinecourses.science.psu.edu/stat501/node/253

Unsupervised Learning

...

Reinforcement Learning

- Sarsa: On-Policy TD Control

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha * [R_{t+1} + \gamma * Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

- Q-learning: O ff-Policy TD Control

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha * [R_{t+1} + \gamma * \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

Time series

- Algorithm requirements:
    - extrapolate patterns outside of the domain of training data
    - derive confidence intervals
- Most common algos Linear regression
    - +Handles different time series
    - +High interpretability
    - -Sensitive to outliers
    - -Narrow confidence intervals Exponential smoothing ARIMA (Autoregressive Integrated Moving Average) Dynamic Linear Model
- Sources
    - datascience.com/blog/time-series-forecasting-machine-learning-differences

Articles

Data Science for Startups: Data Pipelines

- towardsdatascience.com/data-science-for-startups-data-pipelines-786f6746a59a
- low latency
- scalability
- interactive querying
- versioning
- monitoring
- testing