SZTAKI

Stochastic Models & Adaptive Algorithms

Balázs Csanád Csáji

SZTAKI: Institute for Computer Science and Control

13-17 Kende utca, Budapest, Hungary, 1111

Ph.D. Course, BME & ELTE, Semester: 2020/21/1

Main Parts of the Course

I. Statistical Learning (40 %)

From linear regression to kernel methods. Performance guarantees. This part mainly addresses regression and classification problems.

II. Reinforcement Learning (40 %)

Learning via interactions with an uncertain, dynamic environment. Markov decision processes and model-free online learning methods.

III. Stochastic Approximation (20 %)

A general theory of recursive methods working with uncertain data. E.g., stochastic optimization, acceleration techniques, consistency.

Assumed Mathematical Background

Linear Algebra

Positive (semi-) definite matrices, QR decomposition, eigenvalues, eigenvectors, pseudoinverse, matrix norms, range and null space.

Mathematical analysis

Convex functions, gradient vector, Hessian matrix, Lagrange multipliers, Hilbert spaces, Riesz representation theorem.

Probability & Statistics

Central moments, convergence concepts (a.s., in distribution, etc.) bias (of an estimator), Fisher information, statistical efficiency.

*"One must learn by doing the thing;
for though you think you know it,
you have no certainty, until you try."*

(Sophocles)

Numerical Experiments

Part I: Linear Regression

Numerical Exercise I.1

- Generate a sample of size n, $\{(x_i, y_i)\}_{i=1}^{n}$, where $\{x_i\}$ are i.i.d. random variables having a uniform distribution on $\big[0, 1\big]$; let $y_i = f(x_i) + \varepsilon_i$, where $f(x) = x\,\sin(c\,x)$, e.g., with $c = 16$, and the noises, $\{\varepsilon_i\}$, are zero-mean i.i.d. Gaussian random variables.
- Construct linear regression estimates with d basis functions using the least-squares criterion. Experiment with different types of basis functions, such as $f_i(x) = x^{i-1}$ and $f_i(x) = \exp(-\|x-\mu_i\|^2/\sigma_i^2)$; e.g., one can use $\mu_i = (i-1)/(d-1)$ and a fixed constant σ^2 for σ_i^2.
- Calculate the LS estimate with QR decomposition or with SVD.
- Apply the recursive formulation of LS for increasing sample sizes.
- Also make experiments for the least-norm problem, i.e., if $d > n$.
- Plot the observations, the original function, and the resulting regression functions for various sample sizes and basis choices.

Numerical Exercise I.2

– Consider the scalar AR (auto-regressive) time-series model,

$$y_t \;=\; a \cdot y_{t-1} + b \cdot y_{t-2} + \varepsilon_t,$$

where $\{\varepsilon_t\}$ are i.i.d. standard normal; a and b are parameters (constants) with $b < 1 + a$, $b < 1 - a$ and $b > -1$ (for stability).

– This system can be written in a linear regression form as

$$y_t \;=\; \varphi_t^{\mathrm{T}} \theta^* + \varepsilon_t,$$

where $\varphi_t = (y_{t-1}, y_{t-2})^{\mathrm{T}}$, the rows of Φ, and $\theta^* = (a, b)^{\mathrm{T}}$.

– The error of the LS estimates is asymptotically Gaussian, that is

$$\sqrt{n}\,(\hat{\theta}_n - \theta^*) \xrightarrow{d} N(0, \Gamma) \quad \text{as} \quad n \to \infty,$$

where $\hat{\theta}_n$ is the LS estimate for sample size n, "$\xrightarrow{d}$" denotes convergence in distribution and $N(0, \Gamma)$ is the normal distribution.

Numerical Exercise I.2

- The asymptotic covariance matrix Γ is $\sigma_0^2 \, \mathbb{E}\big[\varphi_0 \varphi_0^{\mathrm{T}}\big]$, where σ_0^2 is the variance of the noises, $\{\varepsilon_t\}$ (i.e., one for standard normal), and φ_0 has the stationary distribution of the regressors, $\{\varphi_t\}$.
- Assuming $\sigma_0^2 = 1$, this can be estimated by (cf. ergodic theorem)

$$\sigma_0^2 \, \mathbb{E}\big[\varphi_0 \varphi_0\big] \approx \Gamma_n = \frac{1}{n} \sum_{t=1}^{n} \varphi_t \varphi_t^{\mathrm{T}}.$$

- Choose an appropriate θ^*, some initial conditions (y_0, y_{-1}) and simulate a trajectory of size $n = 100$. Then, plot the LS estimate with its confidence ellipsoids for probabilities $p = 0.9,\ 0.95,\ 0.98$.
- Of course, one should build the ellipsoids by using the fact that

$$n \, (\hat{\theta}_n - \theta^*)^{\mathrm{T}} \, \Gamma_n^{-1} \, (\hat{\theta}_n - \theta^*) \ \dot{\sim} \ \chi^2(\mathrm{dim}(\theta^*)),$$

where $\chi^2(d)$ is the χ^2 distribution with d degrees of freedom.

Requirements

- Perform the experiments using Matlab or Python. Adequately structure and comment your codes, to make them easily readable.
- Use matrix manipulating functions for QR decompositon, SVD, inverse, etc.; but do not use existing functions for pseudoinverse or model fitting, such as the backslash operator of Matlab for LS.
- Write a short, 1-3 pages, report with figures which describes and illustrates your experiments. Make the report in LaTeX or Jupyter.
- Send the codes and the report in email to csaji@sztaki.hu.
- Hint 1: if a square matrix A is badly conditioned, a simple way to decrease its condition number is to replace it with $A + \lambda I$, where I is the identity matrix and λ is a tiny scalar (cf. ridge regression).
- Hint 2: one can look up the inverse CDF values of $\chi^2(d)$ from a table, but there are also functions in Matlab and Python to do it.

*"I hear, I forget;
I see, I remember;
I do, I understand."*

(Chinese proverb)

Numerical Experiments

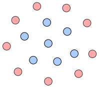
Part II: Kernel Methods

Numerical Exercise II.1

- Consider a binary classification problem on the plane, $\mathbb{R}^2$, where the data are generated as follows: for each observation, for both classes, there is a 0.5 probability of getting the observation from that class; and the classes have multivariate normal distributions.
- The means of the distributions should be different (mandatory), and their covariance matrices could also be different (optional).
- Generate datasets and implement the following linear classifiers: soft margin SVC, LS-SVC and nearest centroid classifier (NCC).
- For the first two cases, use and solve the primal form of the corresponding optimization problems (do not use SVM libraries).
- Plot the datasets and the obtained classification boundaries (for SVC, LS-SVC and NCC) for various regularization parameters (λ) and various datasets (generated by different means/covariances).

Numerical Exercise II.2

- Consider a binary classification problem on the plane, $\mathbb{R}^2$, where one of the classes mostly concentrates in a compact area, while the other class surrounds the previous class (with potential overlaps).



- First, design an algorithm to generate such classification datasets.
- Implement the nonlinear (kernelized) soft margin SVC method using the Gaussian kernel, namely, $k(x, y) = \exp(-\|x - y\|^2/\sigma^2)$.
- For this, the Wolfe dual of soft margin SVC should be solved by a convex optimization tool (do not directly use SVM libraries).
- Plot the datasets and the obtained SVC classification boundaries for various regularization parameters (λ) and kernel widths (σ).

Requirements

- Perform the experiments using Matlab or Python. Adequately structure and comment your codes, to make them easily readable.
- Use a convex optimization library, e.g., cvx in Matlab or cvxpy in Python, to implement the corresponding primal or dual problems.
- Important: do not use existing methods dedicated to SVMs, such as the ones provided by the LibSVM library in Matlab and Python.
- The aim is to implement SVC based on mathematical optimization tools and not to simply use an existing SVM library for that.
- Also, do not copy SVC from a source code found on the internet.
- Write a short, 1-3 pages, report with figures which describes and illustrates your experiments. Preferably, make the report in LaTeX. (you can also make the documentation in Jupyter notebook).
- Send the codes and the report in email to csaji@sztaki.hu.

*"Being a student is easy.
Learning requires actual work."*

(William Crawford)

Numerical Experiments

Part III: Reinforcement Learning

Numerical Exercise III.1

- Consider a small-scale Markov Decision Process (MDP), such as the gambler's problem, cliff walking or even a randomly generated MDP (e.g., see Examples 4.3 and 6.6 of Sutton and Barto's book: "Reinforcement Learning: An Introduction", 2nd. edition, 2018).
- Implement the following model-based MDP solution algorithms: value iteration (VI), policy iteration (PI), linear programming (LP).
- Compute the optimal value function of the selected MDP with LP.
- Then, plot its distance (using Euclidean distance, i.e., $\|\cdot\|_2$) from the value functions calculated by the VI and PI algorithms.
- The comparison should be per iteration (hence, the x-axis of the figure should show the iteration number, while the y-axis should present the distances of VI and PI from the optimal solution).

Numerical Exercise III.2

- Consider the same small-scale MDP as in the first experiments.
- Implement the model-free Q-learning using a totally (uniformly) random, an ε-greedy and a Boltzmann (semi-greedy) base policy.
- Thus, you should simulate the environment using a base policy and iteratively learn the optimal action-value function by applying Watkins' Q-learning on the obtained state-action-cost trajectories.
- Hints: ε could be 0.1 or 0.05, while τ could be 0.1, 1, and 10.
- Let V^* be the optimal value function (computed by LP in the first experiments) and let $V_t(x) \doteq \min_a Q_t(x, a)$ be the value function induced by a Q-learning algorithm (for iteration t). Plot the distances ($\| \cdot \|_2$) of V^* and V_t for the three Q-learning approaches.
- Also plot (on a different figure) the gathered costs/rewards during the learning process for the case of all three base policies above.

Requirements

- Perform the experiments using Matlab or Python. Adequately structure and comment your codes, to make them easily readable.
- Use a convex optimization library, e.g., cvx in Matlab or cvxpy in Python, to implement the resulting linear programming problem.
- Important: it is allowed to use existing RL environments for the MDPs, but not for the solution algorithms. Do not use existing methods dedicated to RL, implement the RL algorithms yourself.
- The aim is to understand the fundamental RL approaches by implementing them, and not simply to apply RL codes of others.
- Write a short, 1-3 pages, report with figures which describes and illustrates your experiments. Preferably, make the report in LaTeX. (you can also make the documentation in Jupyter notebook).
- Send the codes and the report in email to csaji@sztaki.hu.

"*The true method of knowledge is experiment.*"
(William Blake)

Numerical Experiments

Part IV: Stochastic Approximation

Numerical Exercise IV.1

- Consider the Q-learning algorithm implemented in Num. Ex. III.2 (any of the three base policies could be used for this exercise).
- Study the extension of Q-learning with momentum acceleration and Polyak averaging. If γ_n and β_n are the stepsizes of Q-learning and the momentum term, resp., consider using $\gamma_n = c_1/n(x,a)$ and $\beta_n = c_2/n(x,a)$, where $n(x,a)$ is the number of visits to (x,a).
- Important: when calculating the momentum terms and the Polyak averages of the estimates, the $Q_n(x,a)$ values should be treated as different stochastic approximation processes for each (x,a).
- Namely, only those $Q_n(x,a)$ values should be taken into account for the momentum or averages, where that (x,a) pair was visited.
- Compare the standard Q-learning with variants using momentum acceleration (study different c_i's). For both approaches, also make the comparison if one applies Polyak averaging using a fix window.

Requirements

- Perform the experiments using Matlab or Python. Adequately structure and comment your codes, to make them easily readable.
- Use the MDP studied in Numerical Exercise III, therefore, it is allowed to use existing RL environments for the MDP model.
- Extend your implementation of Q-learning from Num. Ex. III.2. Do not use existing RL libraries, implement the methods yourself.
- Write a short, 1-3 pages, report with figures which describes and illustrates your experiments. Preferably, make the report in LaTeX. (you can also make the documentation in Jupyter notebook).
- It should also contain plots of iterations vs distances from V^*, as in Num. Ex. III.2, for the algorithms (Q-learning, QL with momentum, and Polyak averages of both; preferably for various (c_1, c_2) pairs).
- Send the codes and the report in email to csaji@sztaki.hu.

Info for the Exam

Overview of the Key Concepts

Question Groups for the Exam

- Deterministic linear regression: normal equations, orthogonal projections, recursive LS, least-norm problem, pseudoinverse, SVD
- Stochastic linear regression: Gauss-Markov theorem, ML vs LS, strong consistency, limiting distribution, PCA, ridge regression
- Nonlinear optimization: Lagrangian, weak and strong duality, Karush-Kuhn-Tucker conditions, Slater's condition, Wolfe duality
- Statistical learning theory: classification, true vs empirical risk, Bayes classifier, VC dim., ERM vs SRM, hard/soft margin SVM
- Kernel methods: Wolfe dual of SVMs, RKHS, representer theorem
- Markov decision processes: value functions, Bellman operators, value and policy iteration, linear programming, policy gradient
- Reinforcement learning: Monte Carlo, TD(λ), SARSA, Q-learning
- Stochastic approximation: Robbins-Monro, Kiefer-Wolfowitz, SPSA, momentum acceleration, Polyak averaging, Lyapunov functions

Thank you for your attention!

www.sztaki.hu/~csaji csaji@sztaki.hu