Bibliography

[1] Abadi, M., Barham, P., Chen, J., et al.: TensorFlow: A system for large-scale machine learning. In: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation. pp. 265–283. OSDI'16, USENIX Association, Berkeley, CA, USA (2016)

[2] Adler, A., Elad, M., Hel-Or, Y., Rivlin, E.: Sparse coding with anomaly detection. Journal of Signal Processing Systems 79(2), 179–188 (May 2015)

[3] Ahmad, S.: Running swarms (May 2017), `http://nupic.docs.numenta.org/0.6.0/guide-swarming.html`

[4] Ahmed, T., Coates, M., Lakhina, A.: Multivariate online anomaly detection using kernel recursive least squares. In: Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM). pp. 625–633. IEEE (2007)

[5] Alarcon Aquino, V.: Anomaly Detection and Prediction in Communication Networks Using Wavelet Transforms. Ph.D. thesis, Imperial College London (2003)

[6] Alarcon-Aquino, V., Barria, J.A.: Anomaly detection in communication networks using wavelets. IET Proceedings-Communications 148(6), 355–362 (2001)

[7] Alickovic, E., Subasi, A.: Medical decision support system for diagnosis of heart arrhythmia using DWT and random forests classifier. Journal of Medical Systems 40(4), 108 (2016)

[8] Aminikhanghahi, S., Cook, D.J.: A survey of methods for time series change point detection. Knowledge and Information Systems (KAIS) 51(2), 339–367 (2017)

[9] An, J., Cho, S.: Variational autoencoder based anomaly detection using reconstruction probability. Special Lecture on IE 2(1), 1–18 (2015)

[10] Anon.: Auf dem Weg zur Lösung der ICE-Probleme. Eisenbahn-Revue International 10, 446–447 (1998)

[11] Ansmann, G.: Efficiently and easily integrating differential equations with JiTCODE, JiTCDDE, and JiTCSDE. Chaos 28(4), 043116 (2018)

[12] Arpit, D., Zhou, Y., Ngo, H., Govindaraju, V.: Why regularized auto-encoders learn sparse representation? In: Proceedings of the International Conference on Machine Learning (ICML). pp. 136–144. PMLR (2016)

[13] Bai, S., Kolter, J.Z., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. CoRR abs/1803.01271 (2018)

[14] Bergstra, J., et al.: Hyperopt: A python library for model selection and hyperparameter optimization. Computational Science & Discovery 8(1), 014008 (2015)

[15] Bishop, C.M.: Pattern Recognition and Machine Learning. springer (2006)

[16] Blázquez-García, A., Conde, A., Mori, U., Lozano, J.A.: A review on outlier/anomaly detection in time series data. arXiv preprint arXiv:2002.04236 (2020)

[17] Bontempi, G., Ben Taieb, S.: Statistical foundations of machine learning, chapter: Recursive least squares. Université Libre de Bruxelles (2008)

[18] Bontemps, L., McDermott, J., Le-Khac, N.A., et al.: Collective anomaly detection based on long short term memory recurrent neural network. In: Proceedings of the International Conference on Future Data and Security Engineering (FDSE). pp. 141–152. Springer (2016)

[19] Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LOF: Identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. pp. 93–104 (2000)

[20] de Bruijn, B., Nguyen, T.A., Bucur, D., Tei, K.: Benchmark datasets for fault detection and classification in sensor data. In: Proceedings of the 5th International Conference on Sensor Networks (SENSORNETS). pp. 185–195 (2016)

[21] Carpenter, G.A., Grossberg, S.: ART 2: Self-organization of stable category recognition codes for analog input patterns. Applied Optics 26(23), 4919–4930 (1987)

[22] Chakraborty, G., Kamiyama, T., Takahashi, H., Kinoshita, T.: An efficient anomaly detection in quasi-periodic time series data — A case study with ECG. In: Rojas, I., Pomares, H., Valenzuela, O. (eds.) Time Series Analysis and Forecasting, pp. 147–157. Springer International Publishing (2018)

[23] Chalapathy, R., Chawla, S.: Deep learning for anomaly detection: A survey. CoRR abs/1901.03407 (2019)

[24] Chalapathy, R., Menon, A.K., Chawla, S.: Anomaly detection using one-class neural networks. CoRR abs/1802.06360 (2018)

[25] Chan, D.M., Rao, R., Huang, F., Canny, J.F.: GPU accelerated t-distributed stochastic neighbor embedding. Journal of Parallel and Distributed Computing (JPDC) 131, 1–13 (2019)

[26] Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Computing Surveys (CSUR) 41(3), 15 (2009)

[27] Chaovalit, P., Gangopadhyay, A., Karabatis, G., Chen, Z.: Discrete wavelet transform-based time series analysis and mining. ACM Computing Surveys (CSUR) 43(2), 6:1–6:37 (Feb 2011)

[28] Chauhan, S., Vig, L.: Anomaly detection in ECG time signals via deep long short-term memory networks. In: Proceedings of IEEE International Conference on Data Science and Advanced Analytics (DSAA). pp. 1–7. IEEE (2015)

[29] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)

[30] Chollet, F., et al.: Keras. `https://keras.io` (2015)

[31] Cook, A.A., Mısırlı, G., Fan, Z.: Anomaly detection for IoT time-series data: A survey. IEEE Internet of Things Journal (IoT-J) 7(7), 6481–6494 (2019)
[32] Das, S., Wong, W.K., Dietterich, T., Fern, A., Emmott, A.: Incorporating expert feedback into active anomaly discovery. In: 2016 IEEE 16th International Conference on Data Mining (ICDM). pp. 853–858. IEEE (2016)
[33] Dasgupta, D., Forrest, S.: Novelty detection in time series data using ideas from immunology. In: Proceedings of the 5th International Conference on Intelligent Systems (IS). pp. 82–87. Citeseer (1996)
[34] Dasgupta, D., Forrest, S.: An anomaly detection algorithm inspired by the immune system. In: Artificial Immune Systems and Their Applications, pp. 262–277. Springer (1999)
[35] Dau, H.A., Ciesielski, V., Song, A.: Anomaly detection using replicator neural networks trained on examples of one class. In: Asia-Pacific Conference on Simulated Evolution and Learning (SEAL). pp. 311–322. Springer (2014)
[36] Dauphin, Y.N., Fan, A., Auli, M., Grangier, D.: Language modeling with gated convolutional networks. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning (ICML). Proceedings of Machine Learning Research, vol. 70, pp. 933–941. PMLR (2017)
[37] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 248–255. IEEE (2009)
[38] Deutsche Bahn AG: Zug-Check im Vorbeifahren: Wayside-Monitoring. `https://inside.bahn.de/wayside-monitoring/` (2017), last accessed: 30.30.2021
[39] Di Mattia, F., Galeone, P., De Simoni, M., Ghelfi, E.: A survey on GANs for anomaly detection. CoRR abs/1906.11632 (2019)
[40] Dumoulin, V., Visin, F.: A guide to convolution arithmetic for deep learning. CoRR abs/1603.07285 (2016)
[41] Engel, Y., Mannor, S., Meir, R.: The kernel recursive least-squares algorithm. IEEE Transactions on Signal Processing 52(8), 2275–2285 (2004)
[42] Esslinger, V., Kieselbach, R., Koller, R., Weisse, B.: The railway accident of Eschede – technical background. Engineering Failure Analysis 11(4), 515–535 (2004)
[43] expedia.com: ExpediaDotCom/ adaptive-alerting. `https://github.com/ExpediaDotCom/adaptive-alerting` (2020)
[44] Filonov, P., Kitashov, F., Lavrentyev, A.: RNN-based early cyber-attack detection for the Tennessee Eastman process. CoRR abs/1709.02232 (2017)
[45] Filonov, P., Lavrentyev, A., Vorontsov, A.: Multivariate industrial time series with cyber-attack simulation: Fault detection using an LSTM-based predictive data model. CoRR abs/1612.06676 (2016)
[46] Fischer, M., et al.: Anomaly detection on time series: An evaluation of deep learning methods. `https://github.com/KDD-OpenSource/DeepADoTS` (2019)
[47] Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Computing Surveys (CSUR) 46(4), 44:1–44:37 (2014)

[48] Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N.: Convolutional sequence to sequence learning. CoRR abs/1705.03122 (2017)

[49] George, D., Hawkins, J.: Towards a mathematical theory of cortical micro-circuits. PLOS Computational Biology 5(10), e1000532 (2009)

[50] Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. pp. 249–256 (2010)

[51] Goh, J., Adepu, S., Tan, M., Lee, Z.S.: Anomaly detection in cyber physical systems using recurrent neural networks. In: 18th International Symposium on High Assurance Systems Engineering (HASE). pp. 140–145. IEEE (2017)

[52] Goldberger, A.L., Amaral, L.A., Glass, L., Hausdorff, J.M., Ivanov, P.C., Mark, R.G., Mietus, J.E., Moody, G.B., Peng, C.K., Stanley, H.E.: PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation 101(23), e215–e220 (2000)

[53] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative adversarial networks. CoRR abs/1406.2661 (2014)

[54] Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., Schmidhuber, J.: LSTM: A search space odyssey. IEEE Transactions on Neural Networks and Learning Systems (TNNLS) 28(10), 2222–2232 (2017)

[55] Gupta, M., Gao, J., Aggarwal, C.C., Han, J.: Outlier detection for temporal data: A survey. IEEE Transactions on Knowledge and Data Engineering (TKDE) 26(9), 2250–2267 (2013)

[56] Hannun, A.Y., Rajpurkar, P., Haghpanahi, M., Tison, G.H., Bourn, C., Turakhia, M.P., Ng, A.Y.: Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. Nature Medicine 25(1), 65 (2019)

[57] Hariri, S., Kind, M.C., Brunner, R.J.: Extended isolation forest. IEEE Transactions on Knowledge and Data Engineering (TKDE) 33(4), 1479–1489 (2021)

[58] Hawkins, S., He, H., Williams, G.J., Baxter, R.A.: Outlier detection using replicator neural networks. In: Kambayashi, Y., Winiwarter, W., Arikawa, M. (eds.) Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery (DaWaK). Lecture Notes in Computer Science, vol. 2454, pp. 170–180. Springer (2002)

[59] Haykin, S.: Adaptive Filtering Theory. Pearson, 5th Edition (2013)

[60] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016)

[61] He, Y., Zhao, J.: Temporal convolutional networks for anomaly detection in time series. In: Journal of Physics: Conference Series. vol. 1213, p. 042050. IOP Publishing (2019)

[62] Hochreiter, S.: The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6(02), 107–116 (1998)
[63] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation 9(8), 1735–1780 (1997)
[64] Hodge, V., Austin, J.: A survey of outlier detection methodologies. Artificial Intelligence Review 22(2), 85–126 (2004)
[65] Holschneider, M., Kronland-Martinet, R., Morlet, J., Tchamitchian, P.: A real-time algorithm for signal analysis with the help of the wavelet transform. In: Combes, J.M., Grossmann, A., Tchamitchian, P. (eds.) Wavelets, pp. 286–297. Springer Berlin Heidelberg, Berlin, Heidelberg (1990)
[66] Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4700–4708 (2017)
[67] Hundman, K., Constantinou, V., Laporte, C., Colwell, I., Söderström, T.: Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding. In: Guo, Y., Farooq, F. (eds.) Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD). pp. 387–395. ACM (2018)
[68] Hyndman, R.J., Wang, E., Laptev, N.: Large-scale unusual time series detection. In: International Conference on Data Mining Workshop (ICDMW). pp. 1616–1619. IEEE (2015)
[69] Ibidunmoye, O., Metsch, T., Elmroth, E.: Real-time detection of performance anomalies for cloud services. In: International Symposium on Quality of Service (IWQoS). pp. 1–9. IEEE (2016)
[70] Jain, R., Chlamtac, I.: The P2 algorithm for dynamic calculation of quantiles and histograms without storing observations. Communications of the ACM 28(10), 1076–1085 (1985)
[71] Jiang, W., Hong, Y., Zhou, B., He, X.: A GAN-based anomaly detection approach for imbalanced industrial time series. IEEE Access 7, 143608–143619 (2019)
[72] Jones, J., Palmer, L.: An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex. Journal of Neurophysiology 2, 1233–1258 (1987)
[73] Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. In: Annual Meeting of the Association for Computational Linguistics (ACL). pp. 655–665. Baltimore, Maryland (2014)
[74] Kanarachos, S., Mathew, J., Chroneos, A., Fitzpatrick, M.: Anomaly detection in time series data using a combination of wavelets, neural networks and Hilbert transform. In: Proceedings of the International Conference on Information, Intelligence, Systems and Applications (IISA). pp. 1–6 (2015)
[75] Katser, I.D., Kozitsin, V.O.: Skoltech anomaly benchmark (SKAB) (2020)
[76] KDD cup 1999 (1999), http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[77] Keogh, E., Lin, J., Fu, A.: Hot SAX: Efficiently finding the most unusual time series subsequence. In: Fifth IEEE International Conference on Data Mining (ICDM'05). pp. 8–pp. IEEE (2005)

[78] Keogh, E., Lin, J., Lee, S.H., Van Herle, H.: Finding the most unusual time series subsequence: Algorithms and applications. Knowledge and Information Systems (KAIS) 11(1), 1–27 (2007)

[79] Kieu, T., Yang, B., Guo, C., Jensen, C.S.: Outlier detection for time series with recurrent autoencoder ensembles. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI). pp. 2725–2732 (2019)

[80] Kieu, T., Yang, B., Jensen, C.S.: Outlier detection for multidimensional time series using deep neural networks. In: Proceedings of the International Conference on Mobile Data Management (MDM). pp. 125–134. IEEE (2018)

[81] Kim, S.S., Reddy, A.N., Vannucci, M.: Detecting traffic anomalies using discrete wavelet transform. In: International Conference on Information Networking. pp. 951–961. Springer (2004)

[82] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) Proceedings of the 3rd International Conference on Learning Representations (ICLR) (2015)

[83] Kingma, D.P., Welling, M.: Auto-encoding variational Bayes. In: Bengio, Y., LeCun, Y. (eds.) Proceedings of the International Conference on Learning Representations (ICLR) (2014)

[84] Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J.R.R., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., Duerig, T., Ferrari, V.: The open images dataset V4. International Journal of Computer Vision (IJCV) 128(7), 1956–1981 (2020)

[85] Kwon, D., Ko, K., Vannucci, M., Reddy, A.N., Kim, S.: Wavelet methods for the detection of anomalies and their application to network traffic analysis. Quality and Reliability Engineering International (QREI) 22(8), 953–969 (2006)

[86] Lagerholm, M., Peterson, C., Braccini, G., Edenbrandt, L., Sornmo, L.: Clustering ECG complexes using Hermite functions and self-organizing maps. IEEE Transactions on Biomedical Engineering (T-BME) 47(7), 838–848 (2000)

[87] Laptev, N., Amizadeh, S.: Yahoo anomaly detection dataset webscope S5 (2015), `http://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70`

[88] Laptev, N., Amizadeh, S., Flint, I.: Generic and scalable framework for automated time-series anomaly detection. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). p. 1939–1947. Association for Computing Machinery, New York, NY, USA (2015)

[89] Lavin, A., S. Ahmad, S.: Evaluating real-time anomaly detection algorithms – the Numenta anomaly benchmark. In: Proceedings of the International Conference on Machine Learning and Applications (ICMLA) (2015)

[90] Lemos, A.P., Tierra-Criollo, C., Caminhas, W.: ECG anomalies identification using a time series novelty detection technique. In: IV Latin American Congress on Biomedi-

cal Engineering 2007, Bioengineering Solutions for Latin America Health. pp. 65–68. Springer (2007)

[91] Li, D., et al.: Anomaly detection with generative adversarial networks for multivariate time series. CoRR abs/1809.04758 (2018)

[92] Li, D., et al.: MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks. In: Proceedings of the International Conference on Artificial Neural Networks (ICANN). pp. 703–716. Springer (2019)

[93] Li, H., Xu, Z., Taylor, G., Studer, C., Goldstein, T.: Visualizing the loss landscape of neural nets. In: Advances in Neural Information Processing Systems (NIPS). pp. 6389–6399 (2018)

[94] Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the 8th ACM SIGMOD workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD). pp. 2–11 (2003)

[95] Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing SAX: A novel symbolic representation of time series. Data Mining and knowledge discovery (DMKD) 15(2), 107–144 (2007)

[96] Lin, M., Chen, Q., Yan, S.: Network in network. In: Bengio, Y., LeCun, Y. (eds.) Proceedings of the 2nd International Conference on Learning Representations (ICLR) (2014)

[97] LinkedIn: linkedin/luminol. `https://github.com/linkedin/luminol` (2015)

[98] Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: Eighth IEEE International Conference on Data Mining (ICDM). pp. 413–422. IEEE (2008)

[99] Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3431–3440 (2015)

[100] Lu, W., Ghorbani, A.A.: Network anomaly detection based on wavelet analysis. EURASIP Journal on Advances in Signal Processing 2009, 4 (2009)

[101] Luz, E.J.d.S., Schwartz, W.R., Cámara-Chávez, G., Menotti, D.: ECG-based heartbeat classification for arrhythmia detection: A survey. Computer methods and programs in biomedicine 127, 144–164 (2016)

[102] Ma, J., Perkins, S.: Online novelty detection on temporal sequences. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). pp. 613–618 (2003)

[103] Ma, J., Perkins, S.: Time-series novelty detection using one-class support vector machines. In: Proceedings of the International Joint Conference on Neural Networks. vol. 3, pp. 1741–1745. IEEE (2003)

[104] Ma, J., Theiler, J., Perkins, S.: Accurate on-line support vector regression. Neural Computation 15(11), 2683–2703 (2003)

[105] van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. Journal of Machine Learning Research (JMLR) 9, 2579–2605 (2008)

[106] Mackey, M.C., Glass, L.: Oscillation and chaos in physiological control systems. Science 197(4300), 287–289 (1977)

[107] Malhotra, P., Vig, L., Shroff, G., Agarwal, P.: Long short term memory networks for anomaly detection in time series. In: Proceedings of the 23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN). pp. 89–94 (2015)

[108] Malhotra, P., et al.: LSTM-based encoder-decoder for multi-sensor anomaly detection. CoRR abs/1607.00148 (2016)

[109] Markou, M., Singh, S.: Novelty detection: A review – part 1: Statistical approaches. Signal Processing 83(12), 2481–2497 (2003)

[110] Markou, M., Singh, S.: Novelty detection: A review – part 2: Neural network based approaches. Signal Processing 83(12), 2499–2521 (2003)

[111] Meyer, Y., Salinger, D.: Wavelets and Operators, Cambridge Studies in Advanced Mathematics, vol. 1. Cambridge University Press (1995)

[112] Moody, G.B., Mark, R.G.: PhysioNet: The MIT-BIH arrhythmia database. `https://www.physionet.org/physiobank/database/mitdb/` (1992)

[113] Moody, G.B., Mark, R.G.: The impact of the MIT-BIH arrhythmia database. IEEE Engineering in Medicine and Biology Magazine (EMBS) 20(3), 45–50 (2001)

[114] Moya, M.M., Hush, D.R.: Network constraints and multi-objective optimization for one-class classification. Neural Networks 9(3), 463–474 (Apr 1996)

[115] Munir, M., et al.: DeepAnT: A deep learning approach for unsupervised anomaly detection in time series. IEEE Access 7, 1991–2005 (2019)

[116] Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning (ICML). pp. 807–814 (2010)

[117] Nason, G.: wavethresh: Wavelets Statistics and Transforms (2016), `https://CRAN.R-project.org/package=wavethresh`, R package version 4.6.8

[118] Ng, A., et al.: Sparse autoencoder. CS294A Lecture Notes 72(2011), 1–19 (2011)

[119] Nguyen, D., Kefalas, M., Yang, K., Apostolidis, A., Olhofer, M., Limmer, S., Bäck, T.: A review: Prognostics and health management in automotive and aerospace. International Journal of Prognostics and Health Management 10(2), 35 (2019)

[120] Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: Proceedings of the IEEE international conference on computer vision (ICCV). pp. 1520–1528 (2015)

[121] Nolle, F., Badura, F., Catlett, J., Bowser, R., Sketch, M.: CREI-GARD, a new concept in computerized arrhythmia monitoring systems. Computers in Cardiology 13, 515–518 (1986)

[122] Odena, A., Dumoulin, V., Olah, C.: Deconvolution and checkerboard artifacts. Distill 1(10), e3 (2016)

[123] Oh, D.Y., Yun, I.D.: Residual error based anomaly detection using auto-encoder in SMD machine sound. Sensors 18(5) (2018)

[124] van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A.W., Kavukcuoglu, K.: Wavenet: A generative model for raw audio. In: Proceedings of the 9th ISCA Speech Synthesis Workshop. p. 125. ISCA (2016)

[125] Otter, D.W., Medina, J.R., Kalita, J.K.: A survey of the usages of deep learning for natural language processing. IEEE Transactions on Neural Networks and Learning Systems (TNNLS) (2020)

[126] Oxford Dictionary of English: anomaly, n. Oxford University Press, Oxford, 3 edn. (2016), https://www.unilexids.de/

[127] Pan, J., Tompkins, W.J.: A real-time QRS detection algorithm. IEEE Transactions on Biomedical Engineering (T-BME) 32(3), 230–236 (1985)

[128] Pang, G., Shen, C., Cao, L., van den Hengel, A.: Deep learning for anomaly detection: A review. CoRR abs/2007.02500 (2020)

[129] Park, D., Hoshi, Y., Kemp, C.C.: A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder. IEEE Robotics and Automation Letters 3(3), 1544–1551 (2018)

[130] Paszke, A., et al.: PyTorch: An imperative style, high-performance deep learning library. In: Wallach, H., et al. (eds.) Advances in Neural Information Processing Systems (NIPS), pp. 8024–8035. Curran Assoc. (2019)

[131] Patcha, A., Park, J.M.: An overview of anomaly detection techniques: Existing solutions and latest technological trends. Computer Networks 51(12), 3448–3470 (2007)

[132] Pereira, J., Silveira, M.: Learning representations from healthcare time series data for unsupervised anomaly detection. In: Proceedings of the International Conference on Big Data and Smart Computing (BigComp). pp. 1–7 (Feb 2019)

[133] Pereira, J., Silveira, M.: Unsupervised anomaly detection in energy time series data using variational recurrent autoencoders with attention. In: Wani, M.A., et al. (eds.) Proceedings of the International Conference on Machine Learning and Applications (ICMLA). pp. 1275–1282. IEEE (2018)

[134] Pereira, J., Silveira, M.: Unsupervised representation learning and anomaly detection in ECG sequences. International Journal of Data Mining and Bioinformatics 22(4), 389–407 (2019)

[135] Pimentel, M.A., Clifton, D.A., Clifton, L., Tarassenko, L.: A review of novelty detection. Signal Processing 99, 215–249 (2014)

[136] Pimentel, T., Monteiro, M., Veloso, A., Ziviani, N.: Deep active learning for anomaly detection. In: 2020 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2020)

[137] Price, G.R.: Extension of covariance selection mathematics. Annals of Human Genetics 35(4), 485–490 (1972)

[138] Raginsky, M., Willett, R.M., Horn, C., Silva, J., Marcia, R.F.: Sequential anomaly detection in the presence of noise and limited feedback. IEEE Transactions on Information Theory 58(8), 5544–5562 (2012)

[139] Rai, H.M., Trivedi, A., Shukla, S.: ECG signal processing for abnormalities detection using multi-resolution wavelet transform and artificial neural network classifier. Measurement 46(9), 3238–3246 (2013)

[140] Remy, P.: Temporal convolutional networks for Keras. GitHub repository: `https://github.com/philipperemy/keras-tcn` (2020)

[141] Rosner, B.: Percentage points for a generalized ESD many-outlier procedure. Technometrics 25(2), 165–172 (1983)

[142] Rousseeuw, P.J., Driessen, K.V.: A fast algorithm for the minimum covariance determinant estimator. Technometrics 41(3), 212–223 (1999)

[143] Sahoo, S., Kanungo, B., Behera, S., Sabut, S.: Multiresolution wavelet transform based feature extraction and ECG classification to detect cardiac abnormalities. Measurement 108, 55–66 (2017)

[144] Sakurada, M., Yairi, T.: Anomaly detection using autoencoders with nonlinear dimensionality reduction. In: Proceedings of the 2nd Workshop on Machine Learning for Sensory Data Analysis (MLSDA). pp. 4–11 (2014)

[145] Salimans, T., Kingma, D.P.: Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In: Advances in Neural Information Processing Systems (NIPS). pp. 901–909 (2016)

[146] Saurav, S., Malhotra, P., TV, V., Gugulothu, N., Vig, L., Agarwal, P., Shroff, G.: Online anomaly detection with concept drift adaptation using recurrent neural networks. In: Proceedings of the ACM India Joint International Conference on Data Science and Management of Data (CODS-COMAD). pp. 78–87 (2018)

[147] Schölkopf, B., Williamson, R.C., Smola, A.J., Shawe-Taylor, J., Platt, J.C.: Support vector method for novelty detection. In: Advances in Neural Information Processing Systems (NIPS). pp. 582–588 (2000)

[148] Shahabi, C., Chung, S., Safar, M.: A wavelet-based approach to improve the efficiency of multi-level surprise mining. In: PAKDD International Workshop on Mining Spatial and Temporal Data. vol. 2001 (2001)

[149] Shen, L., Li, Z., Kwok, J.: Timeseries anomaly detection using temporal hierarchical one-class network. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems (NeurIPS). vol. 33, pp. 13016–13026. Curran Associates, Inc. (2020)

[150] Sherman, J., Morrison, W.J.: Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. The Annals of Mathematical Statistics 21(1), 124–127 (1950)

[151] Sivaraks, H., Ratanamahatana, C.A.: Robust and accurate anomaly detection in ECG artifacts using time series Motif discovery. Computational and Mathematical Methods in Medicine 2015 (2015)

[152] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research (JMLR) 15(1), 1929–1958 (2014)

[153] Su, Y., Zhao, Y., Niu, C., Liu, R., Sun, W., Pei, D.: Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD). pp. 2828–2837 (2019)

[154] Suh, S., Chae, D.H., Kang, H.G., Choi, S.: Echo-state conditional variational autoencoder for anomaly detection. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN). pp. 1015–1022. IEEE (2016)

[155] Świechowski, M., Park, H., Mańdziuk, J., Kim, K.J.: Recent advances in general game playing. The Scientific World Journal 2015 (2015)

[156] Szegedy, C., et al.: Going deeper with convolutions. CoRR abs/1409.4842 (2014)

[157] Talagala, P.D., Hyndman, R.J., Smith-Miles, K., Kandanaarachchi, S., Muñoz, M.A.: Anomaly detection in streaming nonstationary temporal data. Journal of Computational and Graphical Statistics (JCGS) 29(1), 13–27 (2020)

[158] Tan, S.C., Ting, K.M., Liu, T.F.: Fast anomaly detection for streaming data. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI) (2011)

[159] Tatbul, N., Lee, T.J., Zdonik, S., Alam, M., Gottschlich, J.: Precision and recall for time series. In: Bengio, S., Wallach, H.M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems (NeurIPS). pp. 1924–1934 (2018)

[160] Taylor, M., et al.: numenta/nupic: 1.0.5. `https://doi.org/10.5281/zenodo.1257382` (2018)

[161] Thill, M., Däubener, S., Konen, W., Bäck, T.: Anomaly detection in electrocardiogram readings with stacked LSTM networks. In: ITAT. CEUR Workshop Proceedings, vol. 2473, pp. 17–25 (2019)

[162] Thill, M., Konen, W., Bäck, T.: Anomaly detection in time series with discrete wavelet transforms and maximum likelihood estimation. In: Hoffmann, F., Hüllermeier, E. (eds.) Proceedings 27. Workshop Computational Intelligence (GMA-CI). pp. 67–71. Universitätsverlag Karlsruhe (2017)

[163] Thill, M., Konen, W., Bäck, T.: Online anomaly detection on the webscope S5 dataset: A comparative study. In: Proceedings of the IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS). pp. 1–8. Springer (2017)

[164] Thill, M., Konen, W., Bäck, T.: Time series anomaly detection with discrete wavelet transforms and maximum likelihood estimation. In: Valenzuela, O., Rojas, I., et al. (eds.) Proceedings of the International Conference on Time Series (ITISE). vol. 2, pp. 11–23 (2017)

[165] Thill, M., Konen, W., Bäck, T.: Online adaptable time series anomaly detection with discrete wavelet transforms and multivariate Gaussian distributions. Archives of Data Sciences (AoDSA), Series A (2018)

[166] Thill, M., Konen, W., Bäck, T.: Time series encodings with temporal convolutional networks. In: Filipic, B., Minisci, E.A., Vasile, M. (eds.) Bioinspired Optimization

Methods and Their Applications - 9th International Conference, BIOMA 2020, Brussels, Belgium, November 19-20, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12438, pp. 161–173. Springer (2020)

[167] Thill, M., Konen, W., Bäck, T.: MGAB: The mackey-glass anomaly benchmark (2020)

[168] Thomas, M., Das, M.K., Ari, S.: Automatic ECG arrhythmia classification using dual tree complex wavelet based features. AEU - International Journal of Electronics and Communications 69(4), 715–721 (Apr 2015)

[169] Tsipouras, M.G., Fotiadis, D.I., Sideris, D.: Arrhythmia classification using the RR-interval duration signal. In: Computers in Cardiology. pp. 485–488. IEEE (2002)

[170] Tsymbal, A.: The problem of concept drift: Definitions and related work. Computer Science Department, Trinity College Dublin 106(2), 58 (2004)

[171] Tuncer, T., Dogan, S., Pławiak, P., Acharya, U.R.: Automated arrhythmia detection using novel hexadecimal local pattern and multilevel wavelet transform with ECG signals. Knowledge-Based Systems 186, 104923 (2019)

[172] van Vaerenbergh, S., Santamaría, I., Lázaro-Gredilla, M.: Estimation of the forgetting factor in kernel recursive least squares. In: Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing. pp. 1–6. IEEE (2012)

[173] Vallis, O., Hochenbaum, J., Kejariwal, A.: A novel technique for long-term anomaly detection in the cloud. In: 6th USENIX Workshop on Hot Topics in Cloud Computing, Philadelphia, PA (2014)

[174] Voulodimos, A., Doulamis, N., Doulamis, A., Protopapadakis, E.: Deep learning for computer vision: A brief review. Computational Intelligence and Neuroscience 2018 (2018)

[175] Wang, C., Viswanathan, K., Choudur, L., Talwar, V., Satterfield, W., Schwan, K.: Statistical techniques for online anomaly detection in data centers. In: 12th IFIP/IEEE International Symposium on Integrated Network Management and Workshops. pp. 385–392. IEEE (2011)

[176] Welford, B.P.: Note on a method for calculating corrected sums of squares and products. Technometrics 4(3), 419–420 (1962)

[177] Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning 23(1), 69–101 (1996)

[178] Wilcoxon, F.: Individual comparisons by ranking methods. In: Breakthroughs in Statistics, pp. 196–202. Springer (1992)

[179] Woodbury, M.A.: Inverting modified matrices. Memorandum Report 42, Statistical Research Group, Princeton University, Princeton, NJ, (1950)

[180] Xu, H., et al.: Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in web applications. In: Proceedings of The Web Conference (WWW). pp. 187–196 (2018)

[181] Yao, Y., Sharma, A., Golubchik, L., Govindan, R.: Online anomaly detection for sensor systems: A simple and efficient approach. Performance Evaluation 67(11), 1059–1075 (2010)

[182] Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. In: Proceedings of the 4th International Conference on Learning Representations (ICLR) (2016)
[183] Zhai, S., Cheng, Y., Lu, W., Zhang, Z.: Deep structured energy based models for anomaly detection. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning (ICML). p. 1100–1109. ICML'16, JMLR.org (2016)
[184] Zhang, C., Song, D., Chen, Y., Feng, X., Lumezanu, C., Cheng, W., Ni, J., Zong, B., Chen, H., Chawla, N.V.: A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 1409–1416 (2019)
[185] Zhang, Z.M., Chen, S., Liang, Y.Z.: Baseline correction using adaptive iteratively reweighted penalized least squares. Analyst 135, 1138–1146 (2010)
[186] Zhou, B., Liu, S., Hooi, B., Cheng, X., Ye, J.: BeatGAN: Anomalous rhythm detection using adversarially generated time series. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI). pp. 4433–4439 (2019)
[187] Zhu, L., Laptev, N.: Deep and confident prediction for time series at Uber. In: Proceedings of the IEEE International Conference on Data Mining Workshops (ICDMW). pp. 103–110. IEEE (2017)
[188] Zong, B., et al.: Deep autoencoding Gaussian mixture model for unsupervised anomaly detection. In: Proceedings of the International Conference on Learning Representations (ICLR) (2018)

Appendices

Appendix A

Extended Results for Chapter 7

Table A.1: Summary for the ECG-25 data (mean $\pm\, \sigma_{\text{mean}}$ of 10 runs, except for the deterministic NuPic algorithm). The results shown here are for the sum of TP, FN and FP over all 25 time series. For each algorithm and time series, the anomaly threshold was tuned on 10 % of the data, as described in Section 7.4.2.2.

	TP	FN	FP	Prec	Rec	F1	p
Algorithm							
NuPIC	354.7 ± 5.4	366.3 ± 5.4	2214.5 ± 35.3	0.140 ± 0.003	0.492 ± 0.008	0.217 ± 0.004	1.948e-18
SORAD	566.6 ± 3.8	154.4 ± 3.8	743.2 ± 15.4	0.438 ± 0.006	0.786 ± 0.005	0.561 ± 0.005	1.948e-18
DNN-AE	551.3 ± 3.4	169.7 ± 3.4	672.5 ± 8.4	0.452 ± 0.004	0.765 ± 0.005	0.568 ± 0.004	1.948e-18
LSTM-ED	578.8 ± 3.2	142.2 ± 3.2	652.3 ± 17.6	0.480 ± 0.007	0.803 ± 0.004	0.597 ± 0.005	1.948e-18
TCN-AE (baseline)	640.4 ± 1.9	80.6 ± 1.9	630.9 ± 24.1	0.518 ± 0.008	0.888 ± 0.003	0.650 ± 0.006	2.480e-18
LSTM-AD	569.3 ± 1.9	151.7 ± 1.9	411.8 ± 8.7	0.585 ± 0.005	0.790 ± 0.003	0.671 ± 0.004	9.517e-18
TCN-AE (final)	691.3 ± 0.7	29.7 ± 0.7	352.8 ± 10.2	0.668 ± 0.006	0.959 ± 0.001	0.786 ± 0.004	–

Table A.2: F_1-scores (mean $\pm\, \sigma_{\text{mean}}$) of TCN-AE and the other algorithms (highest values in boldface). Same as Table 7.9, except that we permit the algorithms to determine an anomaly threshold from only 10% of the anomaly labels. In all cases in which we fail to reject the null hypothesis at the 5%-confidence level, we add a gray background to the corresponding field.

	NuPIC		LSTM-ED		DNN-AE		LSTM-AD		TCN-AE
	F1	p	F1	p	F1	p	F1	p	F1
1	0.077±0.012	1.94e-18	0.475±0.012	2.07e-18	0.809±0.016	0.00108	0.801±0.017	0.000857	**0.831±0.014**
2	0.164±0.038	8.63e-15	0.254±0.014	2.37e-10	0.276±0.012	4.85e-10	**0.561±0.014**	1.0	0.467±0.024
3	0.082±0.016	1.95e-18	0.77 ±0.01	4.6e-09	0.729±0.017	9.57e-08	0.345±0.014	1.95e-18	**0.845±0.011**
4	0.106±0.035	6.3e-06	0.112±0.018	1.94e-18	0.144±0.018	1.29e-17	**0.420±0.015**	1.0	0.278±0.031
8	0.294±0.041	1.95e-18	0.861±0.009	8.61e-06	0.538±0.013	7.07e-18	0.789±0.009	1.22e-11	**0.903±0.010**
9	0.108±0.027	2.01e-18	0.471±0.009	6.23e-13	0.544±0.013	0.0103	0.397±0.021	6.39e-11	**0.573±0.011**
10	0.509±0.064	3.79e-15	0.691±0.012	5.3e-06	0.644±0.024	9.3e-13	**0.796±0.011**	0.955	0.785±0.014
11	0.043±0.018	1.17e-18	0.515±0.034	0.0807	0.285±0.028	7.4e-15	**0.640±0.039**	0.967	0.556±0.023
12	0.09 ±0.06	1.8e-18	0.351±0.019	1.27e-15	0.343±0.027	1.25e-14	0.578±0.028	0.381	**0.593±0.022**
13	0.010±0.007	1.33e-18	0.718±0.022	4.99e-08	0.588±0.028	2.25e-15	0.624±0.029	9.26e-14	**0.817±0.022**
14	0.372±0.080	2e-18	0.793±0.015	4.31e-08	0.614±0.011	9.52e-18	0.684±0.019	1.42e-15	**0.869±0.010**
15	0.006±0.003	1.78e-18	0.574±0.018	4.46e-10	0.702±0.023	0.791	**0.781±0.008**	1.0	0.692±0.015
16	0.264±0.019	1.93e-18	0.816±0.010	1.1e-06	0.612±0.008	3.78e-17	0.873±0.010	0.00805	**0.883±0.013**
17	0.001±0.001	1.97e-19	0.097±0.029	4.66e-16	0.10 ±0.03	9.66e-16	0.10 ±0.03	9.66e-16	**0.791±0.029**
18	0.251±0.004	1.95e-18	0.482±0.007	1.95e-18	0.660±0.017	3.06e-18	0.837±0.006	4.07e-14	**0.917±0.005**
20	0.007±0.007	3.74e-18	0.368±0.032	0.000163	0.288±0.032	1.1e-07	0.329±0.036	0.00217	**0.436±0.032**
21	0.004±0.004	1.47e-18	0.361±0.033	2.1e-11	0.230±0.021	2.88e-12	0.391±0.021	0.000737	**0.558±0.038**
22	0.337±0.067	1.75e-17	0.500±0.024	1.1e-07	0.538±0.026	1.81e-05	0.621±0.022	0.212	**0.629±0.018**
23	0.281±0.032	1.94e-18	0.504±0.022	2.74e-17	0.769±0.017	0.000898	0.740±0.016	2.52e-05	**0.830±0.012**
26	0.135±0.023	1.95e-18	0.561±0.024	7.71e-11	0.460±0.019	5.74e-18	0.381±0.013	5.01e-14	**0.665±0.015**
28	0.426±0.038	1.94e-18	0.704±0.019	2.45e-06	0.723±0.020	0.00418	0.775±0.012	0.00466	**0.801±0.010**
33	0.0	1.34e-05	**0.092±0.014**	1.0	0.062±0.007	0.999	0.002±0.001	1.34e-05	0.033±0.007
39	0.150±0.019	1.95e-18	0.719±0.010	1.77e-17	0.781±0.019	2.93e-06	0.823±0.011	6.18e-08	**0.898±0.007**
42	0.468±0.021	1.95e-18	0.738±0.014	3.59e-13	0.736±0.023	6.34e-07	0.714±0.012	6.9e-16	**0.844±0.008**
48	0.008±0.004	1.8e-18	0.544±0.024	1.39e-08	0.498±0.032	2.97e-05	**0.703±0.024**	0.925	0.667±0.022
Σ	0.217±0.012	1.95e-18	0.597±0.005	1.95e-18	0.568±0.004	1.95e-18	0.671±0.004	9.52e-18	**0.786±0.004**
mean	0.168±0.009	1.95e-18	0.523±0.004	1.95e-18	0.507±0.004	1.95e-18	0.588±0.003	3.78e-18	**0.686±0.005**
median	0.120±0.015	1.95e-18	0.544±0.006	1.95e-18	0.531±0.008	1.95e-18	0.650±0.007	1.22e-17	**0.747±0.007**

Table A.3: Impact of the individual TCN-AE components for the ECG-25 Data (mean $\pm\, \sigma_{\text{mean}}$ of 10 runs). The results shown here are for the sum of TP, FN and FP over all 25 time series. For each algorithm variant and time series the anomaly threshold was tuned on 10 different segments containing only 10 % of the data.

Algorithm	TP	FN	FP	Prec	Rec	F1	p
noAnomScoreCorr	588.9 ± 2.6	132.1 ± 2.6	527.7 ± 14.0	0.536 ± 0.007	0.817 ± 0.004	0.645 ± 0.006	2.008078e-18
baseline	640.4 ± 1.9	80.6 ± 1.9	630.9 ± 24.1	0.518 ± 0.008	0.888 ± 0.003	0.650 ± 0.006	2.480521e-18
noSkip	655.6 ± 1.8	65.4 ± 1.8	537.3 ± 19.2	0.563 ± 0.008	0.909 ± 0.002	0.691 ± 0.006	8.453158e-18
noLatent	651.9 ± 1.8	69.1 ± 1.8	522.8 ± 19.5	0.570 ± 0.009	0.904 ± 0.002	0.695 ± 0.007	5.052100e-17
noRecon	678.0 ± 1.3	43.0 ± 1.3	414.2 ± 12.7	0.628 ± 0.007	0.940 ± 0.002	0.751 ± 0.005	1.226155e-10
noInvDil	680.2 ± 1.2	40.8 ± 1.2	417.1 ± 13.7	0.630 ± 0.008	0.943 ± 0.002	0.752 ± 0.005	8.187313e-13
noMapReduc	687.1 ± 0.8	33.9 ± 0.8	381.2 ± 11.1	0.650 ± 0.007	0.953 ± 0.001	0.771 ± 0.005	2.041980e-04
final	691.3 ± 0.7	29.7 ± 0.7	352.8 ± 10.2	0.668 ± 0.006	0.959 ± 0.001	0.786 ± 0.004	0.000000e+00

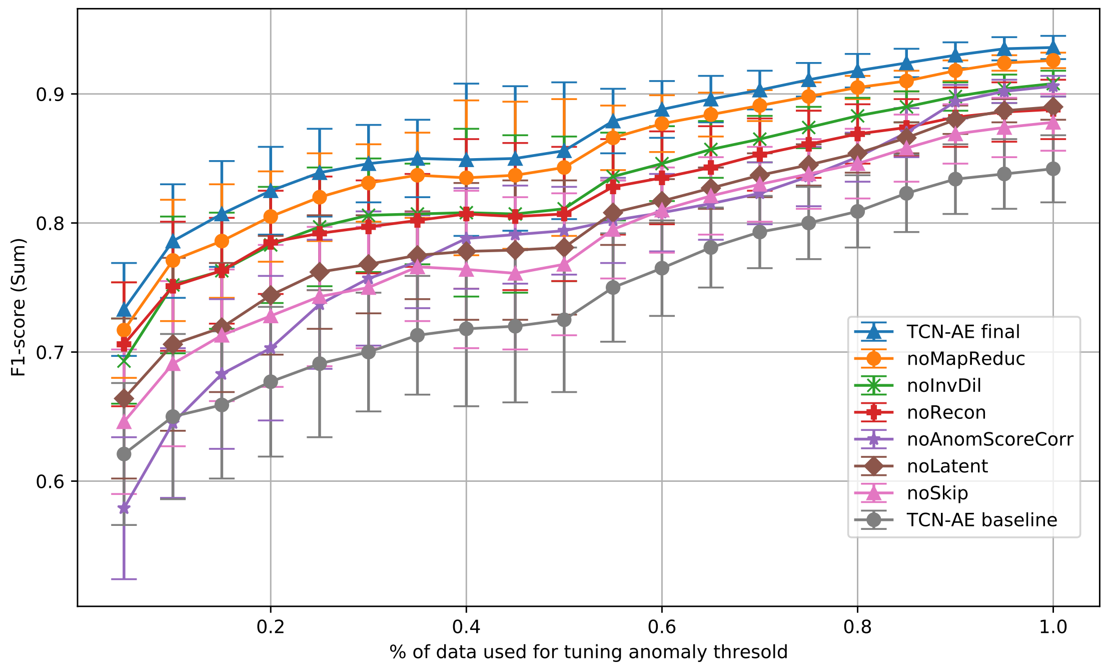

Figure A.1: Results when tuning various algorithmic variants of TCN-AE on different subsets of the ECG-25 data (mean and standard deviation of 10 runs).

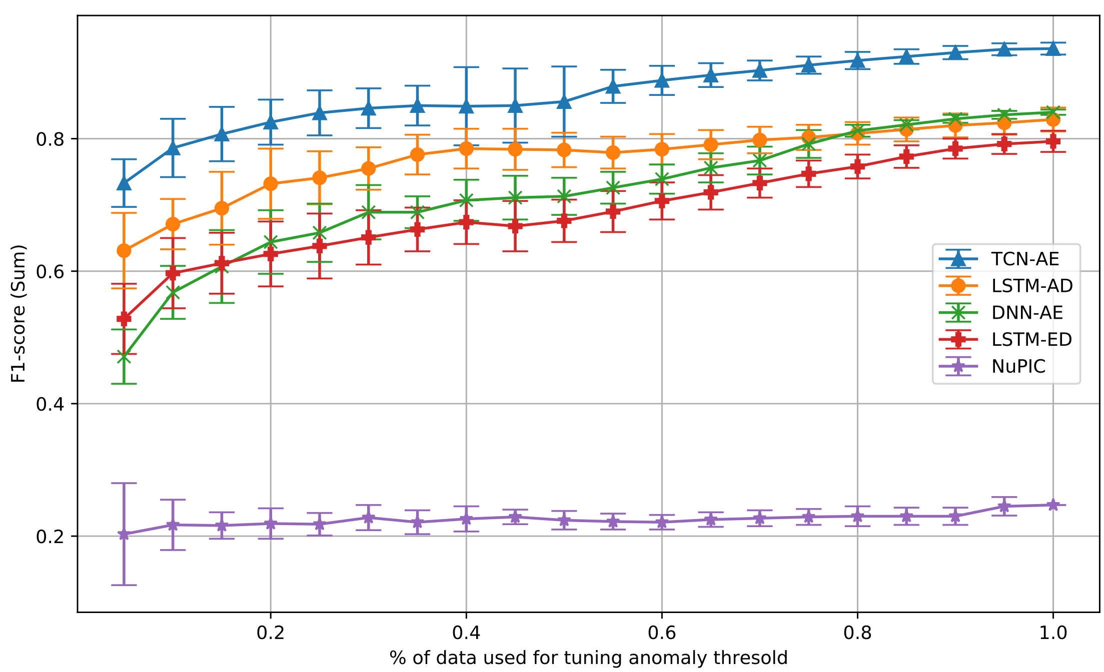

Figure A.2: Results when tuning various algorithms on different subsets of the ECG-25 data (mean and standard deviation of 10 runs).

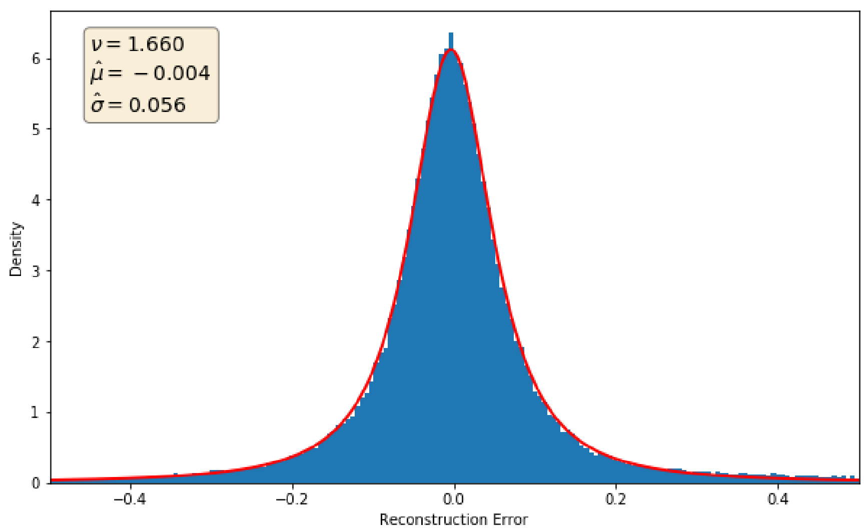

Figure A.3: Histogram of the reconstruction error (in blue) for one dimension of the error matrix $\mathbf{E}'$. We found that the reconstruction errors are bell-shaped (elliptic in higher dimensions). Although the reconstruction errors are not Gaussian, they closely follow a t-distribution with a mean $\hat{\mu}$, a standard deviation $\hat{\sigma}$, and ν degrees of freedom, as indicated by the estimated distribution (red line).

Table A.4: F_1-scores (mean $\pm\,\sigma_{\text{mean}}$) of the individual TCN-AE variants on all 25 time series of the ECG-25 benchmark. The p-values are computed with the one-sided Wilcoxon signed-rank test, in which we compare the final TCN-AE algorithm with the other variants. We compare the F_1-scores of ten runs, which are obtained for an EAC. The Wilcoxon test has the null hypothesis that the median F_1-score of TCN-AE (final) is smaller than the compared algorithm against the alternative that the median F_1-score is larger. In all cases in which we fail to reject the null hypothesis at a confidence level of 5%, we highlight the corresponding field.

Patient No	baseline F1	baseline p	noSkip F1	noSkip p	noLatent F1	noLatent p	noInvDil F1	noInvDil p	noRecon F1	noRecon p	noMapReduc F1	noMapReduc p	noAnomScoreCorr F1	noAnomScoreCorr p	final F1
1	0.768±0.079	0.069	0.844±0.039	0.018	0.940±0.001	0.995	0.913±0.005	0.051	0.846±0.014	0.002	0.937±0.002	0.978	**0.941**	0.995	0.919±0.006
2	0.617±0.043	0.043	0.667	0.042	0.667±0.078	0.168	0.600±0.067	0.035	0.662±0.005	0.046	0.600±0.067	0.035	**0.800±0.022**	0.841	0.733±0.083
3	0.877±0.008	0.003	0.883±0.009	0.002	0.831±0.014	0.002	0.862±0.006	0.002	0.882±0.013	0.002	0.909±0.011	0.041	0.846±0.006	0.002	**0.933±0.008**
4	0.300±0.082	0.002	0.750±0.083	0.013	0.5	0.001	0.5	0.001	0.55 ±0.05	0.001	0.900±0.067	0.079	**1.0**	–	**1.0**
8	0.811±0.016	0.003	0.837±0.015	0.002	0.811±0.012	0.003	0.958±0.004	0.003	0.970±0.004	0.006	0.977±0.002	0.111	0.839±0.015	0.003	**0.981±0.003**
9	0.613±0.045	0.077	**0.675±0.009**	0.556	0.643±0.007	0.007	0.643±0.011	0.025	0.602±0.011	0.004	0.618±0.012	0.007	0.673±0.007	0.556	0.674±0.011
10	0.948±0.016	0.014	0.962±0.007	0.008	0.980±0.005	0.09	0.982±0.003	0.079	0.979±0.004	0.029	**0.987**	–	0.975±0.004	0.023	**0.987**
11	0.9 ±0.1	0.159	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**
12	0.750±0.083	0.013	0.600±0.067	0.002	0.95 ±0.05	0.159	0.650±0.077	0.004	0.600±0.067	0.002	0.750±0.112	0.029	**1.0**	–	**1.0**
13	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**
14	0.903±0.009	0.009	0.938±0.005	0.088	0.938±0.001	0.039	0.938±0.006	0.124	0.922±0.003	0.003	0.939±0.005	0.2	0.924±0.005	0.005	**0.945±0.004**
15	0.894±0.010	0.079	**0.909**	–	0.894±0.010	0.079	**0.909**	–	0.841±0.008	0.001	0.894±0.010	0.079	**0.909**	–	**0.909**
16	0.905±0.009	0.002	0.932±0.002	0.002	0.956±0.005	0.764	0.949±0.002	0.029	0.962±0.001	0.994	0.950±0.001	0.051	**0.982±0.002**	0.998	0.953±0.001
17	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**
18	0.777±0.031	0.003	0.809±0.011	0.003	0.865±0.017	0.002	0.942±0.004	0.002	0.946±0.003	0.002	0.943±0.005	0.006	0.899±0.003	0.003	**0.962±0.003**
20	0.800±0.133	0.079	**1.0**	–	0.967±0.033	0.159	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**
21	0.9 ±0.1	0.159	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**
22	0.867±0.054	0.023	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**	–	**1.0**
23	0.857±0.014	0.002	0.931±0.008	0.017	0.938±0.006	0.032	**0.952**	–	**0.952**	–	**0.952**	–	0.946±0.007	0.159	**0.952**
26	0.638±0.011	0.002	0.671±0.012	0.002	0.646±0.015	0.002	0.694±0.006	0.002	**0.817±0.013**	0.764	0.768±0.015	0.061	0.635±0.016	0.003	0.806±0.011
28	0.874±0.008	0.008	0.912±0.004	–	0.874±0.006	0.005	0.905±0.004	0.179	**0.926±0.005**	0.987	0.906±0.006	0.207	0.924±0.005	0.977	0.912±0.004
33	**0.0**	–	**0.0**	–	**0.0**	–	**0.0**	–	**0.0**	–	**0.0**	–	**0.0**	–	**0.0**
39	0.859±0.012	0.003	0.897±0.010	0.002	0.945±0.003	0.029	0.952±0.003	0.5	0.950±0.003	0.159	**0.955±0.002**	0.921	0.95 ±0.01	0.405	0.952±0.003
42	0.825±0.009	0.002	0.853±0.006	0.002	0.831±0.009	0.002	0.882±0.002	0.002	0.868±0.005	0.002	0.885±0.003	0.004	0.849±0.011	0.003	**0.909±0.006**
48	0.950±0.033	0.958	**1.0**	0.987	0.889±0.039	0.841	**1.0**	0.987	**1.0**	0.987	0.975±0.025	0.949	0.875±0.042	–	0.875±0.042
Σ	0.826±0.007	0.003	0.861±0.008	0.003	0.871±0.007	0.003	0.904±0.003	0.003	0.907±0.004	0.003	0.913±0.002	0.008	0.890±0.005	0.003	**0.926±0.003**
mean	0.785±0.009	0.003	0.843±0.009	0.003	0.843±0.007	0.003	0.849±0.005	0.003	0.851±0.006	0.003	0.874±0.008	0.07	0.879±0.004	0.003	**0.896±0.006**
median	0.868±0.006	0.003	0.911±0.007	0.003	0.919±0.007	0.003	0.939±0.004	0.003	0.939±0.003	0.003	0.950±0.002	0.069	0.934±0.005	0.003	**0.953±0.002**

Appendix B

Derivations

B.1 Batch Incremental Weighted Least Squares Estimator for multivariate Regression Tasks

In practice, the well-known recursive least squares (RLS) filter is often used to learn a linear function in a fully online setting. The RLS filter uses exponentially decaying weights in order to be able to adapt to new concepts. In this section, we derive a similar, however, slightly more complex, incremental version of the weighted least squares estimator which can be used for the SORAD algorithm introduced in Chapter 4. Specifically, we will derive a multivariate variant for batches (having a batch size $\mu \geq 1$) with exponentially decaying weights. This is beneficial in setups which do not have to be fully-online (batch processing usually allows to reduce the computation time if parallelization is supported) or in situations where the amount of data is too large to be processed in a single batch. For example, we use this approach to speed up SORAD for the experiments in Chapter 7.

We start with the original closed-form formulation of the weighted least squares estimator:

$$\boldsymbol{\theta} = \left(\mathbf{X}^\mathsf{T}\mathbf{W}\mathbf{X} + \beta\mathbf{I}\right)^{-1}\mathbf{X}^\mathsf{T}\mathbf{W}\mathbf{y}. \tag{B.1}$$

where $\mathbf{X}$ is a matrix containing n inputs of length k as row-vectors, $\mathbf{W}$ is a diagonal weight matrix, carrying a weight for each of the n observations, $\mathbf{y}$ is a n-dimensional target vector with one value for each input vector (as we will see later, we can easily extend our explications to multi-dimensional outputs, where we would instead use a matrix $\mathbf{Y}$). The term $\beta\mathbf{I}$ (regularization factor and identity matrix) is the so-called regularizer, which is used to prevent overfitting.

Since we have n observations we can also slightly modify our above equation, to later indicate the current iteration:

$$\boldsymbol{\theta}_n = \Big(\overbrace{\underbrace{\mathbf{X}_n^\mathsf{T}\mathbf{W}_n}_{=\mathbf{G}_n}\mathbf{X}_n + \beta\mathbf{I}}^{=\mathbf{A}_n}\Big)^{-1}\overbrace{\underbrace{\mathbf{X}_n^\mathsf{T}\mathbf{W}_n}_{=\mathbf{G}_n}\mathbf{y}_n}^{=\mathbf{b}_n} = \Big(\overbrace{\mathbf{G}_n\mathbf{X}_n + \beta\mathbf{I}}^{=\mathbf{A}_n}\Big)^{-1}\overbrace{\mathbf{G}_n\mathbf{y}_n}^{=\mathbf{b}_n} = \mathbf{A}_n^{-1}\mathbf{b}_n, \tag{B.2}$$

$$\boldsymbol{\theta}_n \in \mathbb{R}^k,\ \mathbf{X}_n \in \mathbb{R}^{n\times k},\ \mathbf{W}_n \in \mathbb{R}^{n\times n},\ \mathbf{I} \in \mathbb{R}^{k\times k},\ \boldsymbol{\beta} \in \mathbb{R},\ \mathbf{y}_n \in \mathbb{R}^n$$

$$\mathbf{G}_n \in \mathbb{R}^{k\times n},\ \mathbf{A}_n \in \mathbb{R}^{k\times k},\ \mathbf{b}_n \in \mathbb{R}^{k}.$$

If now a new batch of μ observation pairs $\mathbf{X}_\mu \in \mathbb{R}^{\mu\times k},\ \mathbf{y}_\mu \in \mathbb{R}^{\mu}$ arrives, some of the above matrices and vectors change as follows (the others remain unchanged):

$$\mathbf{X}_{n+\mu} = \begin{bmatrix}\mathbf{X}_n\\ \mathbf{X}_\mu\end{bmatrix},\ \mathbf{W}_{n+\mu} = \begin{bmatrix}\lambda\mathbf{W}_n & \mathbf{0}\\ \mathbf{0}^\mathsf{T} & \mathbf{W}_\mu\end{bmatrix},\ \mathbf{y}_{n+\mu} = \begin{bmatrix}\mathbf{y}_n\\ \mathbf{y}_\mu\end{bmatrix}, \tag{B.3}$$

where $\mathbf{X}_{n+\mu} \in \mathbb{R}^{(n+\mu)\times k},\ \mathbf{W}_{n+\mu} \in \mathbb{R}^{(n+\mu)\times(n+\mu)},\ \lambda \in \mathbb{R},\ \mathbf{W}_\mu \in \mathbb{R}^{\mu\times\mu},\ \mathbf{y}_{n+\mu} \in \mathbb{R}^{n+\mu}$. λ is a constant with which we can retroactively modify the old weights $\mathbf{W}_n$. Now let us insert the definitions in Eq. (B.3) into Eq. (B.2):

$$\begin{aligned}\boldsymbol{\theta}_{n+\mu} &= \Bigg(\overbrace{\underbrace{\begin{bmatrix}\mathbf{X}_n\\ \mathbf{X}_\mu\end{bmatrix}^\mathsf{T}\begin{bmatrix}\lambda\mathbf{W}_n & \mathbf{0}\\ \mathbf{0}^\mathsf{T} & \mathbf{W}_\mu\end{bmatrix}}_{=\mathbf{G}_{n+\mu}}\begin{bmatrix}\mathbf{X}_n\\ \mathbf{X}_\mu\end{bmatrix} + \beta\mathbf{I}}^{=\mathbf{A}_{n+\mu}}\Bigg)^{-1}\overbrace{\underbrace{\begin{bmatrix}\mathbf{X}_n\\ \mathbf{X}_\mu\end{bmatrix}^\mathsf{T}\begin{bmatrix}\lambda\mathbf{W}_n & \mathbf{0}\\ \mathbf{0}^\mathsf{T} & \mathbf{W}_\mu\end{bmatrix}}_{=\mathbf{G}_{n+\mu}}\begin{bmatrix}\mathbf{y}_n\\ \mathbf{y}_\mu\end{bmatrix}}^{=\mathbf{b}_{n+\mu}}\\ &= \Bigg(\overbrace{\mathbf{G}_{n+\mu}\begin{bmatrix}\mathbf{X}_n\\ \mathbf{X}_\mu\end{bmatrix} + \beta\mathbf{I}}^{=\mathbf{A}_{n+\mu}}\Bigg)^{-1}\overbrace{\mathbf{G}_{n+\mu}\begin{bmatrix}\mathbf{y}_n\\ \mathbf{y}_\mu\end{bmatrix}}^{=\mathbf{b}_{n+\mu}} = \mathbf{A}_{n+\mu}^{-1}\mathbf{b}_{n+\mu},\end{aligned} \tag{B.4}$$

where we identify:

$$\mathbf{G}_{n+\mu} = \begin{bmatrix}\mathbf{X}_n\\ \mathbf{X}_\mu\end{bmatrix}^\mathsf{T}\begin{bmatrix}\lambda\mathbf{W}_n & \mathbf{0}\\ \mathbf{0}^\mathsf{T} & \mathbf{W}_\mu\end{bmatrix} \in \mathbb{R}^{k\times(n+\mu)}, \tag{B.5}$$

$$\mathbf{A}_{n+\mu} = \mathbf{G}_{n+\mu}\begin{bmatrix}\mathbf{X}_n\\ \mathbf{X}_\mu\end{bmatrix} + \beta\mathbf{I} \in \mathbb{R}^{k\times k}, \tag{B.6}$$

$$\mathbf{b}_{n+\mu} = \mathbf{G}_{n+\mu}\begin{bmatrix}\mathbf{y}_n\\ \mathbf{y}_\mu\end{bmatrix} \in \mathbb{R}^{k}. \tag{B.7}$$

Now let us expand equation (B.5):

$$\begin{aligned}\mathbf{G}_{n+\mu} &= \begin{bmatrix}\overbrace{\mathbf{X}_n}^{n\times k}\\ \underbrace{\mathbf{X}_\mu}_{\mu\times k}\end{bmatrix}^\mathsf{T}\begin{bmatrix}\overbrace{\lambda\mathbf{W}_n}^{n\times n} & \overbrace{\mathbf{0}}^{n\times\mu}\\ \underbrace{\mathbf{0}^\mathsf{T}}_{\mu\times n} & \underbrace{\mathbf{W}_\mu}_{\mu\times\mu}\end{bmatrix} = \begin{bmatrix}\overbrace{\mathbf{X}_n^\mathsf{T}}^{k\times n} & \overbrace{\mathbf{X}_\mu^\mathsf{T}}^{k\times\mu}\end{bmatrix}\begin{bmatrix}\overbrace{\lambda\mathbf{W}_n}^{n\times n} & \overbrace{\mathbf{0}}^{n\times\mu}\\ \underbrace{\mathbf{0}^\mathsf{T}}_{\mu\times n} & \underbrace{\mathbf{W}_\mu}_{\mu\times\mu}\end{bmatrix}\\ &= \begin{bmatrix}\lambda\mathbf{X}_n^\mathsf{T}\mathbf{W}_n + \mathbf{X}_\mu^\mathsf{T}\mathbf{0}^\mathsf{T} & \mathbf{X}_n^\mathsf{T}\mathbf{0} + \mathbf{X}_\mu^\mathsf{T}\mathbf{W}_\mu\end{bmatrix} = \begin{bmatrix}\lambda\mathbf{X}_n^\mathsf{T}\mathbf{W}_n & \mathbf{X}_\mu^\mathsf{T}\mathbf{W}_\mu\end{bmatrix}\\ &= \begin{bmatrix}\underbrace{\lambda\mathbf{G}_n}_{k\times n} & \underbrace{\mathbf{X}_\mu^\mathsf{T}\mathbf{W}_\mu}_{k\times\mu}\end{bmatrix} \in \mathbb{R}^{k\times(n+\mu)}.\end{aligned}$$

In the next step, let us evaluate $\mathbf{A}_{n+1}$ from Eq. (B.6):

$$\begin{aligned}
\mathbf{A}_{n+\mu} &= \mathbf{G}_{n+\mu}\begin{bmatrix}\mathbf{X}_n\\ \mathbf{X}_\mu\end{bmatrix} + \beta\mathbf{I} = \begin{bmatrix}\underbrace{\lambda\mathbf{G}_n}_{k\times n} & \underbrace{\mathbf{X}_\mu^\mathsf{T}\mathbf{W}_\mu}_{k\times\mu}\end{bmatrix}\begin{bmatrix}\overbrace{\mathbf{X}_n}^{n\times k}\\ \underbrace{\mathbf{X}_\mu}_{\mu\times k}\end{bmatrix} + \beta\mathbf{I} && \text{(B.8)}\\
&= \lambda\mathbf{G}_n\mathbf{X}_n + \mathbf{X}_\mu^\mathsf{T}\mathbf{W}_\mu\mathbf{X}_\mu + \beta\mathbf{I} = \lambda\underbrace{\mathbf{G}_n\mathbf{X}_n + \beta\mathbf{I}}_{=\mathbf{A}_n} + \mathbf{X}_\mu^\mathsf{T}\mathbf{W}_\mu\mathbf{X}_\mu && \text{(B.9)}\\
&= \lambda\underbrace{\mathbf{A}_n}_{k\times k} + \underbrace{\mathbf{X}_\mu^\mathsf{T}\mathbf{W}_\mu\mathbf{X}_\mu}_{k\times k}, \quad \text{with } \mathbf{A}_0 = \beta\mathbf{I},\ \lambda_0 = 1. && \text{(B.10)}
\end{aligned}$$

Since we have to compute the inverse of $\mathbf{A}_{n+\mu}$, it might be helpful to find an incremental formulation, since the inverse is costly to compute. In this case, the Woodbury matrix identity [179] helps:

$$(\mathbf{A}+\mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1}+\mathbf{VA}^{-1}+\mathbf{U})^{-1}\mathbf{VA}^{-1}. \tag{B.11}$$

This gives us:

$$\begin{aligned}
\mathbf{A}_{n+\mu}^{-1} &= (\lambda\mathbf{A}_n)^{-1} - \overbrace{(\lambda\mathbf{A}_n)^{-1}\mathbf{X}_\mu^\mathsf{T}\big(\mathbf{W}_\mu^{-1} + \mathbf{X}_\mu(\lambda\mathbf{A}_n)^{-1}\mathbf{X}_\mu^\mathsf{T}\big)^{-1}}^{=\boldsymbol{\Delta}_\mu}\mathbf{X}_\mu(\lambda\mathbf{A}_n)^{-1}\\
&= (\lambda\mathbf{A}_n)^{-1} - \boldsymbol{\Delta}_\mu\mathbf{X}_\mu(\lambda\mathbf{A}_n)^{-1}, && \text{(B.12)}
\end{aligned}$$

with:

$$\begin{aligned}
\boldsymbol{\Delta}_\mu &= (\lambda\mathbf{A}_n)^{-1}\mathbf{X}_\mu^\mathsf{T}\big(\mathbf{W}_\mu^{-1} + \mathbf{X}_\mu(\lambda\mathbf{A}_n)^{-1}\mathbf{X}_\mu^\mathsf{T}\big)^{-1} \in \mathbb{R}^{k\times\mu} && \text{(B.13)}\\
&= \mathbf{A}_n^{-1}\mathbf{X}_\mu^\mathsf{T}\big(\lambda\mathbf{W}_\mu^{-1} + \mathbf{X}_\mu\mathbf{A}_n^{-1}\mathbf{X}_\mu^\mathsf{T}\big)^{-1} && \text{(B.14)}
\end{aligned}$$

Then, we expand Eq. (B.7):

$$\begin{aligned}
\mathbf{b}_{n+\mu} &= \mathbf{G}_{n+\mu}\begin{bmatrix}\mathbf{y}_n\\ \mathbf{y}_\mu\end{bmatrix} = \begin{bmatrix}\underbrace{\lambda\mathbf{G}_n}_{k\times n} & \underbrace{\mathbf{W}_\mu\mathbf{X}_\mu}_{k\times 1}\end{bmatrix}\begin{bmatrix}\overbrace{\mathbf{y}_n}^{n\times 1}\\ \underbrace{\mathbf{y}_\mu}_{\mu\times 1}\end{bmatrix} && \text{(B.15)}\\
&= \lambda\mathbf{G}_n\mathbf{y}_n + \mathbf{X}_\mu^\mathsf{T}\mathbf{W}_\mu\mathbf{y}_\mu = \lambda\underbrace{\mathbf{b}_n}_{k\times 1} + \underbrace{\mathbf{X}_\mu^\mathsf{T}\mathbf{W}_\mu\mathbf{y}_\mu}_{k\times 1} && \text{(B.16)}
\end{aligned}$$

Now let us insert the results of (B.12) and (B.16) into Eq. (B.4) and then simplify the expression:

$$
\begin{aligned}
\boldsymbol{\theta}_{n+\mu} &= \mathbf{A}_{n+1}^{-1}\mathbf{b}_{n+1} && \text{(B.17)}\\
&= \Big[(\lambda\mathbf{A}_n)^{-1} - \boldsymbol{\Delta}_\mu\mathbf{X}_\mu(\lambda\mathbf{A}_n)^{-1}\Big]\Big[\lambda\mathbf{b}_n + \mathbf{X}_\mu^\mathsf{T}\mathbf{W}_\mu\mathbf{y}_\mu\Big] && \text{(B.18)}\\
&= \underbrace{\mathbf{A}_n^{-1}\mathbf{b}_n}_{\boldsymbol{\theta}_n} + (\lambda\mathbf{A}_n)^{-1}\mathbf{X}_\mu^\mathsf{T}\mathbf{W}_\mu\mathbf{y}_\mu - \boldsymbol{\Delta}_\mu\mathbf{X}_\mu\underbrace{\mathbf{A}_n^{-1}\mathbf{b}_n}_{\boldsymbol{\theta}_n} - \boldsymbol{\Delta}_\mu\mathbf{X}_\mu(\lambda\mathbf{A}_n)^{-1}\mathbf{X}_\mu^\mathsf{T}\mathbf{W}_\mu\mathbf{y}_\mu && \text{(B.19)}\\
&= \boldsymbol{\theta}_n + \Big[(\lambda\mathbf{A}_n)^{-1} - \boldsymbol{\Delta}_\mu\mathbf{X}_\mu(\lambda\mathbf{A}_n)^{-1}\Big]\mathbf{X}_\mu^\mathsf{T}\mathbf{W}_\mu\mathbf{y}_\mu - \boldsymbol{\Delta}_\mu\mathbf{X}_\mu\boldsymbol{\theta}_n && \text{(B.20)}
\end{aligned}
$$

Although we did a few rearrangements, it seems like Eq. (B.20) cannot be simplified further. However, we can find a more compact solution if we look closer at Eq. (B.13):

$$
\begin{aligned}
\boldsymbol{\Delta}_\mu &= (\lambda\mathbf{A}_n)^{-1}\mathbf{X}_\mu^\mathsf{T}\big(\mathbf{W}_\mu^{-1} + \mathbf{X}_\mu(\lambda\mathbf{A}_n)^{-1}\mathbf{X}_\mu^\mathsf{T}\big)^{-1} && \text{(B.21)}\\
\boldsymbol{\Delta}_\mu\big(\mathbf{W}_\mu^{-1} + \mathbf{X}_\mu(\lambda\mathbf{A}_n)^{-1}\mathbf{X}_\mu^\mathsf{T}\big) &= (\lambda\mathbf{A}_n)^{-1}\mathbf{X}_\mu^\mathsf{T} && \text{(B.22)}\\
\boldsymbol{\Delta}_\mu\mathbf{W}_\mu^{-1} + \boldsymbol{\Delta}_\mu\mathbf{X}_\mu(\lambda\mathbf{A}_n)^{-1}\mathbf{X}_\mu^\mathsf{T} &= (\lambda\mathbf{A}_n)^{-1}\mathbf{X}_\mu^\mathsf{T} && \text{(B.23)}\\
\boldsymbol{\Delta}_\mu\mathbf{W}_\mu^{-1} &= \big[(\lambda\mathbf{A}_n)^{-1} - \boldsymbol{\Delta}_\mu\mathbf{X}_\mu(\lambda\mathbf{A}_n)^{-1}\big]\mathbf{X}_\mu^\mathsf{T} && \text{(B.24)}
\end{aligned}
$$

Interestingly, we can find the RHS of Eq. (B.24) also in Eq. (B.20). If we use above relation, we can therefore simplify (B.20) significantly:

$$
\begin{aligned}
\boldsymbol{\theta}_{n+\mu} &= \boldsymbol{\theta}_n + \boldsymbol{\Delta}_\mu\mathbf{W}_\mu^{-1}\mathbf{W}_\mu\mathbf{y}_\mu - \boldsymbol{\Delta}_\mu\mathbf{X}_\mu\boldsymbol{\theta}_n && \text{(B.25)}\\
&= \boldsymbol{\theta}_n + \boldsymbol{\Delta}_\mu\mathbf{y}_\mu - \boldsymbol{\Delta}_\mu\mathbf{X}_\mu\boldsymbol{\theta}_n && \text{(B.26)}\\
&= \boldsymbol{\theta}_n + \boldsymbol{\Delta}_\mu\Big(\mathbf{y}_\mu - \mathbf{X}_\mu\boldsymbol{\theta}_n\Big) && \text{(B.27)}\\
&= \boldsymbol{\theta}_n + \boldsymbol{\Delta}_\mu\mathbf{e}, && \text{(B.28)}
\end{aligned}
$$

where $\mathbf{e}$ is the prediction error:

$$
\mathbf{e}_\mu = \mathbf{y}_\mu - \mathbf{X}_\mu\boldsymbol{\theta}_n. \tag{B.29}
$$

We can summarize our findings for the univariate case with arbritrary weight matrix W_μ as follows:

$$
\begin{aligned}
\mathbf{e}_\mu &= \mathbf{y}_\mu - \mathbf{X}_\mu\boldsymbol{\theta}_n && \text{(B.30)}\\
\boldsymbol{\Delta}_\mu &= \mathbf{A}_n^{-1}\mathbf{X}_\mu^\mathsf{T}\big(\lambda\mathbf{W}_\mu^{-1} + \mathbf{X}_\mu\mathbf{A}_n^{-1}\mathbf{X}_\mu^\mathsf{T}\big)^{-1} && \text{(B.31)}\\
\boldsymbol{\theta}_{n+\mu} &= \boldsymbol{\theta}_n + \boldsymbol{\Delta}_\mu\mathbf{e}_\mu && \text{(B.32)}\\
\mathbf{A}_{n+\mu}^{-1} &= (\lambda\mathbf{A}_n)^{-1} - \boldsymbol{\Delta}_\mu\mathbf{X}_\mu(\lambda\mathbf{A}_n)^{-1} && \text{(B.33)}
\end{aligned}
$$

where

$$\lambda \in \mathbb{R},\ \mathbf{e}_\mu, \mathbf{y}_\mu, \boldsymbol{\theta}_n, \boldsymbol{\theta}_{n+\mu} \in \mathbb{R}^{\mu},\ \mathbf{X}_\mu \in \mathbb{R}^{\mu\times k},\ \boldsymbol{\Delta}_\mu \in \mathbb{R}^{k\times\mu},\ \mathbf{A}_n, \mathbf{A}_{n+\mu} \in \mathbb{R}^{k\times k},\ \mathbf{W}_\mu \in \mathbb{R}^{n\times n},$$
$$\mathbf{A}_0 = \beta\mathbf{I},\ \lambda_0 = 1,\ \boldsymbol{\theta}_0 = \mathbf{0}.$$

Multivariate Batch Recursive Least Squares Algorithm Extending the above equations to the multivariate case with m dimensions is straightforward. We simply have to replace some vectors with matrices. Furthermore, for a simple setup with exponentially decaying weights we set $\lambda \in [0,1]$ and $W_\mu = \mathbf{I}$. Hence, the final multivariate batch RLS algorithm can be described as follows:

$$\mathbf{E}_\mu = \mathbf{Y}_\mu - \mathbf{X}_\mu \boldsymbol{\Theta}_n \tag{B.34}$$
$$\boldsymbol{\Delta}_\mu = \mathbf{A}_n^{-1}\mathbf{X}_\mu^{\mathsf{T}}\big(\lambda\mathbf{I} + \mathbf{X}_\mu\mathbf{A}_n^{-1}\mathbf{X}_\mu^{\mathsf{T}}\big)^{-1} \tag{B.35}$$
$$\boldsymbol{\Theta}_{n+\mu} = \boldsymbol{\Theta}_n + \boldsymbol{\Delta}_\mu\mathbf{E}_\mu \tag{B.36}$$
$$\mathbf{A}_{n+\mu}^{-1} = \frac{1}{\lambda}\mathbf{A}_n^{-1} - \frac{1}{\lambda}\boldsymbol{\Delta}_\mu\mathbf{X}_\mu\mathbf{A}_n^{-1} \tag{B.37}$$

where

$$\lambda \in [0,1],\ \mathbf{E}_\mu, \mathbf{Y}_\mu, \boldsymbol{\Theta}_n, \boldsymbol{\Theta}_{n+\mu} \in \mathbb{R}^{\mu\times m},\ \mathbf{X}_\mu \in \mathbb{R}^{\mu\times k},\ \boldsymbol{\Delta}_\mu \in \mathbb{R}^{k\times\mu},\ \mathbf{A}_n^{-1}, \mathbf{A}_{n+\mu}^{-1} \in \mathbb{R}^{k\times k},$$
$$\mathbf{A}_0^{-1} = \frac{\lambda}{\beta}\mathbf{I},\ \boldsymbol{\Theta}_0 = \mathbf{0}.$$

B.2 Online Estimation of the Sample Mean and Covariance

This section derives several formulas to incrementally compute the weighted mean and the weighted covariance matrix for a multivariate data set. This property is especially useful in online settings, where an algorithm constantly has to update its estimates and in situations where one expects the mean and the covariance matrix to drift over time. We present a fully online procedure and a procedure that can process mini-batches. The derived formulas are used in Chapter 5 for the DWT-MLEAD algorithm and in Chapter 7, for the SORAD algorithm, which was introduced earlier in Chapter 4.

B.2.1 The Weighted Mean and Covariance Matrix

In some cases, one might want to compute the weighted arithmetic mean and a weighted sample covariance (matrix) where particular sample points should contribute more to the final mean (covariances) than others. For example, it could be reasonable to give larger weight to more recent data points if the statistics (e.g., mean) of the underlying distribution change over time (concept drifts). In such cases, we would want the estimator to track the data generating distribution changes and forget about older knowledge. But also other scenarios are possible, in which a weighting of the data points might be useful. In this section, we define the weighted arithmetic mean and weighted sample covariance, explore a few properties related to these weighted statistics, and then design an estimator for the sample mean and covariance exhibiting some forgetting.

In general, the weighted arithmetic mean is defined quite straightforward as:

$$\bar{\mathbf{x}}_n = \frac{\sum_{i=1}^{n} w_i' \mathbf{x}_i}{\sum_{i=1}^{n} w_i'}, \tag{B.38}$$

where $\mathbf{x}_i$ is the i-th data point (vector) and w_i' is the (unnormalized) weight assigned to the corresponding data point. If the weights are normalized to sum 1, then we get:

$$\bar{\mathbf{x}}_n = \sum_{i=1}^{n} w_i \mathbf{x}_i, \tag{B.39}$$

where the normalized weights are defined as:

$$w_i = \frac{w_i'}{W_n} = \frac{w_i'}{\sum_{i=1}^{n} w_i'}, \tag{B.40}$$

with the normalization factor

$$W_n = \sum_{i=1}^{n} w_i'.$$

The special case with $w_i = \frac{1}{n}$ results in the formulation of the conventional mean. According to [137], the biased weighted covariance matrix can be written similarly as:

$$\bar{\mathbf{\Sigma}} = \frac{\bar{\mathbf{M}}^{(n)}}{W_n} = \frac{\sum_{i=1}^{n} w_i'(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\mathsf{T}}{\sum_{i=1}^{n} w_i'}, \tag{B.41}$$

where $\bar{\mathbf{M}}^{(n)}$ is the so called weighted scatter matrix, with

$$\bar{\mathbf{M}}^{(n)} = \sum_{i=1}^{n} w_i'(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\mathsf{T}.$$

If the weights are frequency weights (each weight represents the count of a data point), then an unbiased estimator is found to be [137]:

$$\bar{\mathbf{\Sigma}} = \frac{\bar{\mathbf{M}}^{(n)}}{W_n - 1}. \tag{B.42}$$

Otherwise, the following unbiased estimator can be used [137]:

$$\bar{\mathbf{\Sigma}} = \frac{\bar{\mathbf{M}}^{(n)}}{W_n - W_n^{(2)}/W_n}, \tag{B.43}$$

where

$$W_n^{(2)} = \sum_{i=1}^{n} (w_i')^2. \tag{B.44}$$

B.2.1.1 Incremental (Batch) Estimation of the Weighted Mean and Covariance

It might become necessary to estimate the mean and covariance matrix of a distribution incrementally in practice. This could be the case if one operates on streaming data or has to process large amounts of data, which cannot be handled in one pass. In the general case, we want to handle batch sizes of $\mu \in \mathbb{N}^+$. For a fully online algorithm, we have the extreme case $\mu = 1$. If a new batch of size μ arrives, we have update our old estimates $\bar{\mathbf{x}}_{n-\mu}$ and $\bar{\mathbf{\Sigma}}_{n-\mu}$. For this purpose we have to process the new examples from $k = n - \mu + 1$ until n. We first note a recursive update rule for $\bar{\mathbf{x}}_n$:

$$\begin{aligned}
\bar{\mathbf{x}}_n &= \frac{W_{n-\mu}\bar{\mathbf{x}}_{n-\mu} + \sum\limits_{i=k}^{n} w_i'\mathbf{x}_i}{W_n} && \text{(B.45)}\\
&= \frac{\left(W_n - \sum\limits_{i=k}^{n} w_i'\right)\bar{\mathbf{x}}_{n-\mu} + \sum\limits_{i=k}^{n} w_i'\mathbf{x}_i}{W_n}\\
&= \bar{\mathbf{x}}_{n-\mu} + \frac{-\sum\limits_{i=k}^{n} w_i'\bar{\mathbf{x}}_{n-\mu} + \sum\limits_{i=k}^{n} w_i'\mathbf{x}_i}{W_n}\\
&= \bar{\mathbf{x}}_{n-\mu} + \frac{\sum\limits_{i=k}^{n} w_i'(\mathbf{x}_i - \bar{\mathbf{x}}_{n-\mu})}{W_n},\\
&= \bar{\mathbf{x}}_{n-\mu} + \sum_{i=k}^{n} \frac{w_i'}{W_n}\mathbf{\Delta}_i && \text{(B.46)}
\end{aligned}$$

with

$$\begin{aligned}
k &= n - \mu + 1 && \text{(B.47)}\\
\mathbf{\Delta}_i &= \mathbf{x}_i - \bar{\mathbf{x}}_{n-\mu} && \text{(B.48)}
\end{aligned}$$

Then, we derive an batch update rule for the estimator for the weighted covariance matrix $\bar{\mathbf{\Sigma}}_n$. We start with the weighted scatter matrix:

$$\begin{aligned}
\bar{\mathbf{M}}^{(n)} &= \sum_{i=1}^{n} w'_i(\mathbf{x}_i - \bar{\mathbf{x}}_n)(\mathbf{x}_i - \bar{\mathbf{x}}_n)^\intercal \\
&= \sum_{i=1}^{n} w'_i\big[\mathbf{x}_i\mathbf{x}_i^\intercal - \mathbf{x}_i\bar{\mathbf{x}}_n^\intercal - \bar{\mathbf{x}}_n\mathbf{x}_i^\intercal + \bar{\mathbf{x}}_n\bar{\mathbf{x}}_n^\intercal\big] \\
&= \sum_{i=1}^{n} w'_i\big[\mathbf{x}_i\mathbf{x}_i^\intercal - 2\mathbf{x}_i\bar{\mathbf{x}}_n^\intercal + \bar{\mathbf{x}}_n\bar{\mathbf{x}}_n^\intercal\big] \\
&= \sum_{i=1}^{n} w'_i\mathbf{x}_i\mathbf{x}_i^\intercal - 2\left(\sum_{i=1}^{n} w'_i\mathbf{x}_i\right)\bar{\mathbf{x}}_n^\intercal + \sum_{i=1}^{n} w'_i\bar{\mathbf{x}}_n\bar{\mathbf{x}}_n^\intercal \\
&= \sum_{i=1}^{n} w'_i\mathbf{x}_i\mathbf{x}_i^\intercal - 2W_n\bar{\mathbf{x}}_n\bar{\mathbf{x}}_n^\intercal + W_n\bar{\mathbf{x}}_n\bar{\mathbf{x}}_n^\intercal \\
&= \sum_{i=1}^{n} w'_i\mathbf{x}_i\mathbf{x}_i^\intercal - W_n\bar{\mathbf{x}}_n\bar{\mathbf{x}}_n^\intercal
\end{aligned}$$

The increment of the weighted scatter matrix from index $k = n-\mu+1$ to n can be computed as follows:

$$\begin{aligned}
\mathbf{\Delta}\bar{\mathbf{M}}^{(n)} &= \bar{\mathbf{M}}^{(n)} - \bar{\mathbf{M}}^{(n-\mu)} \\
&= \sum_{i=1}^{n} w'_i\mathbf{x}_i\mathbf{x}_i^\intercal - W_n\bar{\mathbf{x}}_n\bar{\mathbf{x}}_n^\intercal - \sum_{i=1}^{n-\mu} w'_i\mathbf{x}_i\mathbf{x}_i^\intercal + W_{n-\mu}\bar{\mathbf{x}}_{n-\mu}\bar{\mathbf{x}}_{n-\mu}^\intercal \\
&= \sum_{i=k}^{n} w'_i\mathbf{x}_i\mathbf{x}_i^\intercal - W_n\bar{\mathbf{x}}_n\bar{\mathbf{x}}_n^\intercal + W_{n-\mu}\bar{\mathbf{x}}_{n-\mu}\bar{\mathbf{x}}_{n-\mu}^\intercal
\end{aligned} \tag{B.49}$$

With the already known expression from Eq. (B.45) and with a rearranged formulation of Eq. (B.45)

$$\bar{\mathbf{x}}_{n-\mu} = \frac{W_n\bar{\mathbf{x}}_n - \sum\limits_{i=k}^{n} w'_i\mathbf{x}_i}{W_{n-\mu}} \tag{B.50}$$

we can start simplifying Eq. (B.49). We insert Eq. (B.45) and Eq. (B.50) into Eq. (B.49):

$$\mathbf{\Delta}\bar{\mathbf{M}}^{(n)} = \sum_{i=k}^{n} w'_i\mathbf{x}_i\mathbf{x}_i^\intercal - W_n\frac{W_{n-\mu}\bar{\mathbf{x}}_{n-\mu} + \sum\limits_{i=k}^{n} w'_i\mathbf{x}_i}{W_n}\bar{\mathbf{x}}_n^\intercal + W_{n-\mu}\bar{\mathbf{x}}_{n-\mu}\left(\frac{W_n\bar{\mathbf{x}}_n - \sum\limits_{i=k}^{n} w'_i\mathbf{x}_i}{W_{n-\mu}}\right)^\intercal$$

$$
\begin{aligned}
&= \sum_{i=k}^{n} w_i' \mathbf{x}_i \mathbf{x}_i^{\mathsf{T}} - \left(\sum_{i=k}^{n} w_i' \mathbf{x}_i \right) \bar{\mathbf{x}}_n^{\mathsf{T}} - W_{n-\mu} \bar{\mathbf{x}}_{n-\mu} \bar{\mathbf{x}}_n^{\mathsf{T}} + W_n \bar{\mathbf{x}}_{n-\mu} \bar{\mathbf{x}}_n^{\mathsf{T}} - \bar{\mathbf{x}}_{n-\mu} \left(\sum_{i=k}^{n} w_i' \mathbf{x}_i \right)^{\mathsf{T}} \\
&= \sum_{i=k}^{n} w_i' \mathbf{x}_i \mathbf{x}_i^{\mathsf{T}} - \sum_{i=k}^{n} w_i' \mathbf{x}_i \bar{\mathbf{x}}_n^{\mathsf{T}} + \big(W_n - W_{n-\mu} \big) \bar{\mathbf{x}}_{n-\mu} \bar{\mathbf{x}}_n^{\mathsf{T}} - \bar{\mathbf{x}}_{n-\mu} \sum_{i=k}^{n} w_i' \mathbf{x}_i^{\mathsf{T}} \\
&= \sum_{i=k}^{n} w_i' \mathbf{x}_i \mathbf{x}_i^{\mathsf{T}} - \sum_{i=k}^{n} w_i' \mathbf{x}_i \bar{\mathbf{x}}_n^{\mathsf{T}} + \left(\sum_{i=k}^{n} w_i' \right) \bar{\mathbf{x}}_{n-\mu} \bar{\mathbf{x}}_n^{\mathsf{T}} - \bar{\mathbf{x}}_{n-\mu} \sum_{i=k}^{n} w_i' \mathbf{x}_i^{\mathsf{T}} \\
&= \sum_{i=k}^{n} \left(w_i' \mathbf{x}_i \mathbf{x}_i^{\mathsf{T}} - w_i' \mathbf{x}_i \bar{\mathbf{x}}_n^{\mathsf{T}} + w_i' \bar{\mathbf{x}}_{n-\mu} \bar{\mathbf{x}}_n^{\mathsf{T}} - w_i' \bar{\mathbf{x}}_{n-\mu} \mathbf{x}_i^{\mathsf{T}} \right) \\
&= \sum_{i=k}^{n} \left(w_i' \mathbf{x}_i \big(\mathbf{x}_i^{\mathsf{T}} - \bar{\mathbf{x}}_n^{\mathsf{T}} \big) - w_i' \bar{\mathbf{x}}_{n-\mu} \big(\mathbf{x}_i^{\mathsf{T}} - \bar{\mathbf{x}}_n^{\mathsf{T}} \big) \right) \\
&= \sum_{i=k}^{n} w_i' \big(\mathbf{x}_i - \bar{\mathbf{x}}_{n-\mu} \big) \big(\mathbf{x}_i - \bar{\mathbf{x}}_n \big)^{\mathsf{T}} \\
&= \sum_{i=k}^{n} w_i' \boldsymbol{\Delta}_i \big(\mathbf{x}_i - \bar{\mathbf{x}}_n \big)^{\mathsf{T}} \qquad \text{(B.51)}
\end{aligned}
$$

In summary, with equations (B.40), (B.44), (B.48), (B.46), (B.51), and (B.41) we can update our estimates for a new batch (samples from $k = n - \mu + 1$ to n) with:

$$W_n = W_{n-\mu} + \sum_{i=k}^{n} w_i' \qquad \text{(B.52)}$$

$$W_n^{(2)} = W_{n-\mu}^{(2)} + \sum_{i=k}^{n} (w_i')^2 \qquad \text{(B.53)}$$

$$\boldsymbol{\Delta}_i = \mathbf{x}_i - \bar{\mathbf{x}}_{n-\mu} \qquad \text{(B.54)}$$

$$\bar{\mathbf{x}}_n = \bar{\mathbf{x}}_{n-\mu} + \sum_{i=k}^{n} \frac{w_i'}{W_n} \boldsymbol{\Delta}_i \qquad \text{(B.55)}$$

$$\bar{\mathbf{M}}^{(n)} = \bar{\mathbf{M}}^{(n-\mu)} + \sum_{i=k}^{n} w_i' \boldsymbol{\Delta}_i \big(\mathbf{x}_i - \bar{\mathbf{x}}_n \big)^{\mathsf{T}} \qquad \text{(B.56)}$$

$$\bar{\boldsymbol{\Sigma}}_n = \frac{\bar{\mathbf{M}}^{(n)}}{W_n}. \qquad \text{(B.57)}$$

Note that Eq. (B.57) is a biased estimate of the covariance matrix. For an unbiased estimate one can use either Eq. (B.42) or Eq. (B.43). If all weights are set to $w_i' = 1$, we obtain the update rules for the unweighted case.

B.2.2 Incremental Estimation with Exponentially Decaying Weights

If we use constant batch sizes μ and also keep the weights w'_i among a batch constant, an useful incremental algorithm can be retrieved for a particular set of weights, where each weight is

$$w'_i = \lambda^{\lfloor \frac{n-i}{\mu} \rfloor} = \lambda^{M-j},$$

where $\lambda \in (0, 1]$ is the decay rate, $M = n/\mu$ $(\mu \mid n)$ represents the number of batches for n examples, and $j = \lceil i/\mu \rceil$ is the index of the batch for the i-th example. With such a weighting, each example in the most recent batch will be weighted with $w'_k = \cdots = w'_{n-1} = w'_n = 1$, the penultimate batch will have the weights $w'_{n-2\mu+1} = \cdots = w'_{n-\mu-1} = w'_{n-\mu} = \lambda$ and so forth. Ultimately, the older batches will fade away exponentially, for $\lambda < 1$. The advantage of this approach is that we can usually prevent numerical overflows in W_n and the weighted scatter matrix $\bar{\mathbf{M}}^{(n)}$ and that we can adapt our estimates to new concepts in non-stationary environments.

The normalization factor W_n can be computed with:

$$W_n = \sum_{i=1}^{n} w'_i = \sum_{j=1}^{M} \mu\lambda^{M-j} = \mu \frac{1 - \lambda^M}{1 - \lambda}.$$

Note that the weightings $w'_{n-\mu}, w'_{n-\mu-1}, \ldots, w'_1$ of all previous examples $\mathbf{x}_{n-\mu}, \mathbf{x}_{n-\mu-1}, \ldots, \mathbf{x}_1$, as well as the normalization factor W_n, have to be adjusted if a new example $\mathbf{x}_n$ arrives. In this case, each weight $\{w'_i \mid i \leq n - \mu\}$ has to be multiplied with λ and the weights of the new examples are set to $w'_k = \cdots = w'_n = 1$. How does this change the estimation of the mean and the covariance matrix? Typically, if all weights are changed, one would have to re-compute both statistics from scratch. However, we can show that in this particular case, this is not necessary, as described in the following explications: First let us find an update rule for the normalization factor W_n. We can see that $W_{n-\mu}$ is given by:

$$W_{n-\mu} = \mu \sum_{j=1}^{M-1} \lambda^{M-1-j}. \tag{B.58}$$

Then we re-write W_n:

$$\begin{aligned} W_n &= \mu \sum_{j=1}^{M} \lambda^{M-j} \\ &= \mu \sum_{j=1}^{M-1} \lambda^{M-j} + \mu\lambda^0 = \lambda \cdot \mu \sum_{j=1}^{M-1} \lambda^{M-1-j} + \mu \end{aligned}$$

$$= \lambda \cdot W_{n-\mu} + \mu \tag{B.59}$$

Accordingly, for $W_n^{(2)}$we obtain:

$$W_n^{(2)} = \lambda^2 \cdot W_{n-\mu}^{(2)} + \mu.$$

Then, we find an recursive formulation of the estimated mean $\bar{\mathbf{x}}_n$. According to Eq. (B.38), we find:

$$\bar{\mathbf{x}}_n = \frac{\sum_{j=1}^{M} \lambda^{M-j} \sum_{i=k_j}^{n_j} \mathbf{x}_i}{\mu \sum_{j=1}^{M} \lambda^{M-j}} = \frac{\sum_{j=1}^{M} \lambda^{M-j} \boldsymbol{\Sigma}_j}{W_n}, \tag{B.60}$$

where

$$\begin{aligned} n_j &= j\mu\ ,\ \ k_j = n_j - \mu + 1 = \mu(j-1) + 1, \\ \boldsymbol{\Sigma}_j &= \sum_{i=k_j}^{n_j} \mathbf{x}_i. \end{aligned}$$

Then, with

$$\bar{\mathbf{x}}_{n-\mu} = \frac{\sum_{j=1}^{M-1} \lambda^{M-1-j} \boldsymbol{\Sigma}_j}{W_{n-\mu}}, \quad \text{and} \quad W_{n-\mu} = \frac{W_n - \mu}{\lambda},$$

we can obtain a recursive formulation of Eq. (B.60):

$$\begin{aligned} \bar{\mathbf{x}}_n &= \frac{\sum_{j=1}^{M} \lambda^{M-j} \boldsymbol{\Sigma}_j}{W_n} \\ &= \frac{\sum_{j=1}^{M-1} \lambda^{M-j} \boldsymbol{\Sigma}_j + \lambda^0 \boldsymbol{\Sigma}_j}{W_n} = \frac{\lambda \sum_{j=1}^{M-1} \lambda^{M-1-j} \boldsymbol{\Sigma}_j}{W_n} + \frac{\boldsymbol{\Sigma}_M}{W_n} \\ &= \lambda \frac{W_n - \mu}{\lambda} \frac{1}{W_n} \bar{\mathbf{x}}_{n-\mu} + \frac{\boldsymbol{\Sigma}_M}{W_n} = \bar{\mathbf{x}}_{n-\mu} - \frac{\mu \bar{\mathbf{x}}_{n-\mu}}{W_n} + \frac{\boldsymbol{\Sigma}_M}{W_n} \\ &= \bar{\mathbf{x}}_{n-\mu} + \frac{\sum_{i=k}^{n} \mathbf{x}_i - \sum_{i=k}^{n} \bar{\mathbf{x}}_{n-\mu}}{W_n} = \bar{\mathbf{x}}_{n-\mu} + \frac{\sum_{i=k}^{n} \left(\mathbf{x}_i - \bar{\mathbf{x}}_{n-\mu}\right)}{W_n} \\ &= \bar{\mathbf{x}}_{n-\mu} + \frac{\sum_{i=k}^{n} \boldsymbol{\Delta}_i}{W_n} \end{aligned} \tag{B.61}$$

Finally, we only need a to find a recursive form for the weighted scatter matrix $\bar{\mathbf{M}}^{(n)}$. By expanding the recursion in (B.56) we get (with $k = n - \mu + 1$):

$$
\begin{aligned}
\bar{\mathbf{M}}^{(n)} &= \sum_{j=1}^{M} \lambda^{M-j} \sum_{i=k_j}^{n_j} \boldsymbol{\Delta}_i \big(\mathbf{x}_i - \bar{\mathbf{x}}_n\big)^{\mathsf{T}} , \quad n_j = j\mu \ , \ k_j = \mu(j-1)+1 \\
&= \sum_{j=1}^{M-1} \lambda^{M-j} \sum_{i=k_j}^{n_j} \boldsymbol{\Delta}_i \big(\mathbf{x}_i - \bar{\mathbf{x}}_n\big)^{\mathsf{T}} + \lambda^0 \sum_{i=k}^{n} \boldsymbol{\Delta}_i \big(\mathbf{x}_i - \bar{\mathbf{x}}_n\big)^{\mathsf{T}} \\
&= \lambda \sum_{j=1}^{M-1} \lambda^{M-1-j} \sum_{i=k_j}^{n_j} \boldsymbol{\Delta}_i \big(\mathbf{x}_i - \bar{\mathbf{x}}_n\big)^{\mathsf{T}} + \sum_{i=k}^{n} \boldsymbol{\Delta}_i \big(\mathbf{x}_i - \bar{\mathbf{x}}_n\big)^{\mathsf{T}} \\
&= \lambda \bar{\mathbf{M}}^{(n-\mu)} + \sum_{i=k}^{n} \boldsymbol{\Delta}_i \big(\mathbf{x}_i - \bar{\mathbf{x}}_n\big)^{\mathsf{T}},
\end{aligned} \tag{B.62}
$$

since

$$
\bar{\mathbf{M}}^{(n-\mu)} = \sum_{j=1}^{M-1} \lambda^{M-1-j} \sum_{i=k_j}^{n_j} \boldsymbol{\Delta}_i \big(\mathbf{x}_i - \bar{\mathbf{x}}_n\big)^{\mathsf{T}}.
$$

B.2.2.1 Batch and Online Estimation of the Inverse Covariance Matrix

In practice, it is often required to estimate the inverse of the covariance matrix of a sample, for example, when a Mahalanobis distance between points has to be computed. In a fully online setting it can be quite expensive to compute the inverse of the covariance matrix again for each new point arriving, since the complexity of the inverse of a $m \times m$ matrix is about $\mathcal{O}(m^3)$ (slightly less in some highly optimized algorithms). As we will see in the following, it is not necessary to estimate the covariance matrix, if only its inverse is needed and furthermore, the inverse can be adapted incrementally in an efficient manner. The results are shown for the general case with a weighted exponentially decaying estimator.

For a batch-setup ($\mu > 1$), one can use the Woodbury matrix identity [179] to find a recursive definition of the inverse of the scatter matrix $\bar{\mathbf{M}}$ and covariance matrix $\bar{\boldsymbol{\Sigma}}$. The Woodbury matrix identity is given as:

$$
(\mathbf{A} + \mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1} + \mathbf{VA}^{-1} + \mathbf{U})^{-1}\mathbf{VA}^{-1}. \tag{B.63}
$$

First, we write Eq. (B.62) in vectorized form (to be consistent in the notation later, we move the superscript in $\bar{\mathbf{M}}$ to the index):

$$
\bar{\mathbf{M}}_n = \lambda \bar{\mathbf{M}}_{n-\mu} + \sum_{i=k}^{n} \boldsymbol{\Delta}_i \big(\mathbf{x}_i - \bar{\mathbf{x}}_n\big)^{\mathsf{T}}
$$

$$= \lambda \bar{\mathbf{M}}_{n-\mu} + \mathbf{D}_n^\intercal \mathbf{I} \boldsymbol{\mathcal{X}}_n, \tag{B.64}$$

where $k = n - \mu + 1$ and

$$\begin{aligned}
\mathbf{D}_n &= \begin{pmatrix} \boldsymbol{\Delta}_k & \boldsymbol{\Delta}_{k+1} & \cdots & \boldsymbol{\Delta}_n \end{pmatrix}^\intercal \\
\boldsymbol{\mathcal{X}}_n &= \begin{pmatrix} \mathbf{x}_k - \bar{\mathbf{x}}_n & \mathbf{x}_{k+1} - \bar{\mathbf{x}}_n & \cdots & \mathbf{x}_n - \bar{\mathbf{x}}_n \end{pmatrix}^\intercal.
\end{aligned}$$

Then we identify:

$$\mathbf{A} = \lambda \bar{\mathbf{M}}_{n-\mu},\ \mathbf{U} = \mathbf{D}_n^\intercal,\ \mathbf{C} = \mathbf{I},\ \mathbf{V} = \boldsymbol{\mathcal{X}}_n,$$

and find:

$$\begin{aligned}
\bar{\mathbf{M}}_n^{-1} &= \frac{1}{\lambda} \bar{\mathbf{M}}_{n-\mu}^{-1} - \frac{1}{\lambda^2} \bar{\mathbf{M}}_{n-\mu}^{-1} \mathbf{D}_n^\intercal \Big(\mathbf{I}^{-1} + \frac{1}{\lambda} \boldsymbol{\mathcal{X}}_n \bar{\mathbf{M}}_{n-\mu}^{-1} \mathbf{D}_n^\intercal \Big)^{-1} \boldsymbol{\mathcal{X}}_n \bar{\mathbf{M}}_{n-\mu}^{-1} \\
&= \frac{1}{\lambda} \bar{\mathbf{M}}_{n-\mu}^{-1} - \frac{1}{\lambda} \bar{\mathbf{M}}_{n-\mu}^{-1} \mathbf{D}_n^\intercal \Big(\lambda \mathbf{I} + \boldsymbol{\mathcal{X}}_n \bar{\mathbf{M}}_{n-\mu}^{-1} \mathbf{D}_n^\intercal \Big)^{-1} \boldsymbol{\mathcal{X}}_n \bar{\mathbf{M}}_{n-\mu}^{-1}.
\end{aligned}$$

Since we also have to compute an inverse here, it only makes sense to use the Woodbury matrix identity if the batch size μ is significantly smaller than the dimension of the examples $\mathbf{x}_i$. Similarly, in order to compute the inverse fully online ($\mu = 1$), we simply apply the Sherman-Morrison formula [150] – a special case of the Woodbury matrix identity [179] – to incrementally update $\bar{\mathbf{M}}_n^{-1}$. The formula is given by

$$(\mathbf{A} + \mathbf{u}\mathbf{v}^\intercal)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^\intercal\mathbf{A}^{-1}}{1 + \mathbf{v}^\intercal\mathbf{A}^{-1}\mathbf{u}}. \tag{B.65}$$

If we look at Eq. (B.62), we can identify:

$$\mathbf{A} = \lambda \bar{\mathbf{M}}^{(n-1)},\ \mathbf{u} = \boldsymbol{\Delta}_n,\ \mathbf{v} = \mathbf{x}_n - \bar{\mathbf{x}}_n$$

This then leads to:

$$\bar{\mathbf{M}}_n^{-1} = \frac{1}{\lambda} \bar{\mathbf{M}}_{n-1}^{-1} - \frac{\frac{1}{\lambda} \bar{\mathbf{M}}_{n-1}^{-1} \boldsymbol{\Delta}_n (\mathbf{x}_n - \bar{\mathbf{x}}_n)^\intercal \bar{\mathbf{M}}_{n-1}^{-1}}{\lambda + (\mathbf{x}_n - \bar{\mathbf{x}}_n)^\intercal \bar{\mathbf{M}}_{n-1}^{-1} \boldsymbol{\Delta}_n}. \tag{B.66}$$

With Eq. (B.83), we can finally compute the inverse of the covariance matrix with

$$\bar{\boldsymbol{\Sigma}}_n^{-1} = W_n \bar{\mathbf{M}}_n^{-1}. \tag{B.67}$$

In summary, we can write down the following rules for the recursive (iterative) estimation of the mean vector and the covariance matrix, which should be processed in the given order:

$$W_n = \lambda \cdot W_{n-\mu} + \mu \tag{B.68}$$

$$W_n^{(2)} = \lambda^2 \cdot W_{n-\mu}^{(2)} + \mu \tag{B.69}$$
$$\boldsymbol{\Delta}_i = \mathbf{x}_i - \bar{\mathbf{x}}_{n-\mu} \tag{B.70}$$
$$\bar{\mathbf{x}}_n = \bar{\mathbf{x}}_{n-\mu} + \frac{\sum_{i=k}^{n} \boldsymbol{\Delta}_i}{W_n} \tag{B.71}$$
$$\mathbf{D}_n = \begin{pmatrix} \boldsymbol{\Delta}_k & \boldsymbol{\Delta}_{k+1} & \cdots & \boldsymbol{\Delta}_n \end{pmatrix}^\mathsf{T} \tag{B.72}$$
$$\boldsymbol{\mathcal{X}}_n = \begin{pmatrix} \mathbf{x}_k - \bar{\mathbf{x}}_n & \mathbf{x}_{k+1} - \bar{\mathbf{x}}_n & \cdots & \mathbf{x}_n - \bar{\mathbf{x}}_n \end{pmatrix}^\mathsf{T} \tag{B.73}$$
$$\bar{\mathbf{M}}_n = \lambda \bar{\mathbf{M}}_{n-\mu} + \mathbf{D}_n^\mathsf{T} \boldsymbol{\mathcal{X}}_n \tag{B.74}$$
$$\bar{\mathbf{M}}_n^{-1} = \frac{1}{\lambda}\bar{\mathbf{M}}_{n-\mu}^{-1} - \frac{1}{\lambda}\bar{\mathbf{M}}_{n-\mu}^{-1}\mathbf{D}_n^\mathsf{T}\left(\lambda\mathbf{I} + \boldsymbol{\mathcal{X}}_n\bar{\mathbf{M}}_{n-\mu}^{-1}\mathbf{D}_n^\mathsf{T}\right)^{-1}\boldsymbol{\mathcal{X}}_n\bar{\mathbf{M}}_{n-\mu}^{-1} \tag{B.75}$$
$$\bar{\boldsymbol{\Sigma}}_n = \frac{\bar{\mathbf{M}}_n}{W_n}, \quad \bar{\boldsymbol{\Sigma}}_n^{-1} = W_n \bar{\mathbf{M}}_n^{-1}. \tag{B.76}$$

where μ is the batch size and $k = n - \mu + 1$ is the first index in the new batch. Again, for an unbiased estimate of $\bar{\boldsymbol{\Sigma}}$ one should either use Eq. (B.42) or Eq. (B.43).

For a fully online estimation (batch size $\mu = 1$, $k = n$), above update rules simplify to:

$$W_n = \lambda W_{n-1} + 1 \tag{B.77}$$
$$W_n^{(2)} = \lambda^2 W_{n-1} + 1 \tag{B.78}$$
$$\boldsymbol{\Delta}_n = \mathbf{x}_n - \bar{\mathbf{x}}_{n-1} \tag{B.79}$$
$$\bar{\mathbf{x}}_n = \bar{\mathbf{x}}_{n-1} + \frac{\boldsymbol{\Delta}_n}{W_n} \tag{B.80}$$
$$\bar{\mathbf{M}}_n = \lambda \bar{\mathbf{M}}_{n-1} + \boldsymbol{\Delta}_n (\mathbf{x}_n - \bar{\mathbf{x}}_n)^\mathsf{T} \tag{B.81}$$
$$\bar{\mathbf{M}}_n^{-1} = \frac{1}{\lambda}\bar{\mathbf{M}}_{n-1}^{-1} - \frac{\frac{1}{\lambda}\bar{\mathbf{M}}_{n-1}^{-1}\boldsymbol{\Delta}_n(\mathbf{x}_n - \bar{\mathbf{x}}_n)^\mathsf{T}\bar{\mathbf{M}}_{n-1}^{-1}}{\lambda + (\mathbf{x}_n - \bar{\mathbf{x}}_n)^\mathsf{T}\bar{\mathbf{M}}_{n-1}^{-1}\boldsymbol{\Delta}_n} \tag{B.82}$$
$$\bar{\boldsymbol{\Sigma}}_n = \frac{1}{W_n}\bar{\mathbf{M}}_n, \quad \bar{\boldsymbol{\Sigma}}_n^{-1} = W_n \bar{\mathbf{M}}_n^{-1}. \tag{B.83}$$

In Sec. B.2.4, we show that the memory of the fully online estimator is approximately:

$$n_{mem} \approx \frac{1+\lambda}{1-\lambda}.$$

B.2.3 The Covariance of Weighted Sample Means

In this section, we will derive a formula to compute the covariance of the weighted sample means. This formula will be important in the following section to roughly approximate the memory of the estimator for the sample mean and covariance matrix with exponentially decaying weights.

Typically, when one computes the conventional sample mean, the covariances of the sample mean vector $\bar{\mathbf{x}}_n$ (note: here we are talking about the covariance matrix for the sample mean $\bar{\mathbf{x}}_n$ and not the estimated covariance matrix of the sample itself. This is somewhat similar to the computation of the standard error of the mean.) will tend to zero as the sample size n grows larger. Similarly to the standard error of the mean, which can be expressed as

$$\sigma^2_{\bar{X}} = \frac{\sigma^2}{n}, \tag{B.84}$$

the expected covariance matrix $\mathbf{\Sigma}_{\bar{X}}$ is

$$\mathbf{\Sigma}_{\bar{X}} = \frac{1}{n}\mathbf{\Sigma}, \tag{B.85}$$

which indicates, that for a sufficiently large sample n, the estimated mean $\bar{\mathbf{x}}_n$ will most likely be relatively close (under the typical conditions) to the real mean μ_X of the underlying distribution.

However, if the weighted sample means and covariances are computed, this is no longer necessarily the case. As we will see later, in the weighted case, the elements in the covariance matrix of the sample mean will not converge towards zero in certain situations, implying that the sample mean will not converge to the real mean. Some variance will remain in the estimation, and increasing the sample size will not change this. In such cases, we can say that the estimator has a "limited memory". This may be undesirable when estimating stationary distributions on the one hand. On the other hand, a limited memory can make sense if certain statistics of the underlying distribution (e.g., the means or variances) change over time (often called concept drift or concept change), since the estimator can then forget about the past and learn to adapt its parameters to the drifting distribution. To understand the effects that weighted sample statistics generate, we investigate in this section how the covariance of weighted means behave in different settings.

Let us first derive a formulation for the covariance of two weighted means $\bar{X}_n$ and $\bar{Y}_n$, which are computed for the two jointly distributed random variables X and Y. The covariance of two jointly distributed random variables is defined as:

$$\begin{aligned} \mathrm{cov}(X, Y) &= \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])] \\ &= \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]. \end{aligned}$$

Then, the covariance of the two sample means $\bar{X}_n$ and $\bar{Y}_n$ is:

$$\mathrm{cov}(\bar{X}_n, \bar{Y}_n) = \mathrm{cov}\left(\frac{\sum_{i=1}^n w_i' X_i}{\sum_{i=1}^n w_i'}, \frac{\sum_{i=1}^n w_i' Y_i}{\sum_{i=1}^n w_i'}\right)$$

$$= \operatorname{cov}\left(\frac{1}{W_n}\sum_{i=1}^{n} w_i' X_i, \frac{1}{W_n}\sum_{i=1}^{n} w_i' Y_i\right) \tag{B.86}$$

with random variables X_i and Y_i and where

$$W_n = \sum_{i=1}^{n} w_i'.$$

Let us assume that all weights are already normalized so that

$$w_i = \frac{w_i'}{W_n} = \frac{w_i'}{\sum_{i=1}^{n} w_i'}.$$

Then, Eq. (B.86), simplifies to:

$$\begin{aligned}
\operatorname{cov}(\bar{X}_n, \bar{Y}_n) &= \operatorname{cov}\left(\sum_{i=1}^{n} w_i X_i, \sum_{i=1}^{n} w_i Y_i\right) \\
&= \mathbb{E}\left[\sum_{i=1}^{n} w_i X_i \cdot \sum_{j=1}^{n} w_j Y_j\right] - \mathbb{E}[\bar{X}_n]\mathbb{E}[\bar{Y}_n] \qquad \text{(B.87)}
\end{aligned}$$

Let the expected values $\mu_{\bar{X}_n} = \mathbb{E}[\bar{X}_n]$ and $\mu_{\bar{Y}_n} = \mathbb{E}[\bar{Y}_n]$ be the real means of the two jointly distributed random variables, so that Eq. (B.87) can be written as:

$$\begin{aligned}
\operatorname{cov}(\bar{X}_n, \bar{Y}_n) &= \mathbb{E}\left[\sum_{i=1}^{n} w_i X_i \cdot \sum_{j=1}^{n} w_j Y_j\right] - \mu_{\bar{X}_n}\mu_{\bar{Y}_n} \\
&= \mathbb{E}\left[\sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j X_i Y_j\right] - \mu_{\bar{X}_n}\mu_{\bar{Y}_n} \\
&= \mathbb{E}\left[\sum_{i=1}^{n} w_i^2 X_i Y_j + \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} w_i w_j X_i Y_j\right] - \mu_{\bar{X}_n}\mu_{\bar{Y}_n} \qquad \text{(B.88)}
\end{aligned}$$

Note that in above Eq. (B.88), the original sum was split into two, so that the paired (mutually dependent) data points (X_i, Y_i) share the same sum. All other pairs $\{(X_i, Y_j) \mid i \neq j\}$ are statistically independent and can be treated differently in the following. With the relation

$$\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y],$$

for independent random variables X and Y, we get for Eq. (B.88):

$$\begin{aligned}
\operatorname{cov}(\bar{X}_n, \bar{Y}_n) &= \mathbb{E}\left[\sum_{i=1}^{n} w_i^2 X_i Y_j + \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} w_i w_j X_i Y_j\right] - \mu_{\bar{X}_n}\mu_{\bar{Y}_n}\\
&= \mathbb{E}\left[\sum_{i=1}^{n} w_i^2 X_i Y_j\right] + \mathbb{E}\left[\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} w_i w_j X_i Y_j\right] - \mu_{\bar{X}_n}\mu_{\bar{Y}_n}\\
&= \sum_{i=1}^{n} w_i^2 \mathbb{E}[X_i Y_j] + \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} w_i w_j \mathbb{E}[X_i Y_j] - \mu_{\bar{X}_n}\mu_{\bar{Y}_n}\\
&= \sum_{i=1}^{n} w_i^2 \mathbb{E}[X_i Y_j] + \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} w_i w_j \mathbb{E}[X_i]\mathbb{E}[Y_j] - \mu_{\bar{X}_n}\mu_{\bar{Y}_n}.
\end{aligned} \tag{B.89}$$

Since $\mathbb{E}[X_i] = \mu_{X_i} = \mu_{\bar{X}_n}$ and $\mathbb{E}[Y_i] = \mu_{Y_i} = \mu_{\bar{Y}_n}$ (the expected value of the sample mean is the same as the expected value of each element of the sample), we can continue with

$$\begin{aligned}
\operatorname{cov}(\bar{X}_n, \bar{Y}_n) &= \sum_{i=1}^{n} w_i^2 \mathbb{E}[X_i Y_j] + \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} w_i w_j \mathbb{E}[X_i]\mathbb{E}[Y_j] - \mu_{\bar{X}_n}\mu_{\bar{Y}_n}\\
&= \sum_{i=1}^{n} w_i^2 \mathbb{E}[X_i Y_j] + \mu_{\bar{X}_n}\mu_{\bar{Y}_n}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} w_i w_j - \mu_{\bar{X}_n}\mu_{\bar{Y}_n}\\
&= \sum_{i=1}^{n} w_i^2 \left[\operatorname{cov}(X_i, Y_i) + \mu_{\bar{X}_i}\mu_{\bar{Y}_i}\right] + \mu_{\bar{X}_n}\mu_{\bar{Y}_n}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} w_i w_j - \mu_{\bar{X}_n}\mu_{\bar{Y}_n}\\
&= \sum_{i=1}^{n} w_i^2 \left[\operatorname{cov}(X_i, Y_i) + \mu_{\bar{X}_n}\mu_{\bar{Y}_n}\right] + \mu_{\bar{X}_n}\mu_{\bar{Y}_n}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} w_i w_j - \mu_{\bar{X}_n}\mu_{\bar{Y}_n}\\
&= \sum_{i=1}^{n} w_i^2 \operatorname{cov}(X_i, Y_i) + \sum_{i=1}^{n} w_i^2 \mu_{\bar{X}_n}\mu_{\bar{Y}_n} + \mu_{\bar{X}_n}\mu_{\bar{Y}_n}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} w_i w_j - \mu_{\bar{X}_n}\mu_{\bar{Y}_n}\\
&= \sum_{i=1}^{n} w_i^2 \operatorname{cov}(X_i, Y_i) + \underbrace{\mu_{\bar{X}_n}\mu_{\bar{Y}_n}\sum_{i=1}^{n} w_i^2 + \mu_{\bar{X}_n}\mu_{\bar{Y}_n}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} w_i w_j}_{\text{both sums can be merged again}} - \mu_{\bar{X}_n}\mu_{\bar{Y}_n}
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{n} w_i^2 \mathrm{cov}(X_i, Y_i) + \mu_{\bar{X}_n}\mu_{\bar{Y}_n} \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j - \mu_{\bar{X}_n}\mu_{\bar{Y}_n} \\
&= \sum_{i=1}^{n} w_i^2 \mathrm{cov}(X_i, Y_i) + \mu_{\bar{X}_n}\mu_{\bar{Y}_n} \sum_{i=1}^{n} w_i \underbrace{\sum_{j=1}^{n} w_j}_{=1} - \mu_{\bar{X}_n}\mu_{\bar{Y}_n} \\
&= \sum_{i=1}^{n} w_i^2 \mathrm{cov}(X_i, Y_i) + \mu_{\bar{X}_n}\mu_{\bar{Y}_n} \underbrace{\sum_{i=1}^{n} w_i}_{=1} - \mu_{\bar{X}_n}\mu_{\bar{Y}_n} \\
&= \sum_{i=1}^{n} w_i^2 \mathrm{cov}(X_i, Y_i) + \mu_{\bar{X}_n}\mu_{\bar{Y}_n} - \mu_{\bar{X}_n}\mu_{\bar{Y}_n} \\
&= \sum_{i=1}^{n} w_i^2 \mathrm{cov}(X_i, Y_i)
\end{aligned}
\tag{B.90}
$$

Finally, since all pairs (X_i, Y_i) are independent and identically distributed (IID), we can write:

$$
\begin{aligned}
\mathrm{cov}(\bar{X}_n, \bar{Y}_n) &= \sum_{i=1}^{n} w_i^2 \mathrm{cov}(X_i, Y_i) \\
&= \mathrm{cov}(X, Y) \sum_{i=1}^{n} w_i^2.
\end{aligned}
\tag{B.91}
$$

This simple relation that we finally derived in Eq. (B.91) will help us later to determine the "memory" of exponentially decaying weighted estimators.

B.2.4 Memory of the Exponentially Decaying Estimator

Due to the exponentially decaying weights, the estimator described in Sec. B.2.2 has a limited historical memory since older observations fade out more and more with every new data point. With such an approach, it is possible to adapt the parameters to drifting (changing) distributions, however, at the cost of less accuracy when the data generating process is a stationary distribution. For example, this means that for forgetting factors $\lambda < 1$, the (co-) variances of the mean vector do not converge to zero for large sample sizes n. There will always be some fixed amount of noise left in the estimation, depending on the "rate" of forgetting (specified by λ). In this section, we attempt to answer the following question:
What is the memory n_{mem} of an estimator with exponentially decaying weights? Hence: For which sample size n_{mem}, when computing the ordinary mean (without forgetting) instead, can we obtain the same distribution parameters $\mu_{\bar{X}}$ and $\mathbf{\Sigma}_{\bar{X}}$? In other words, we are looking

for a corresponding ordinary estimator that only considers the last n_{mem} data points to compute the mean.

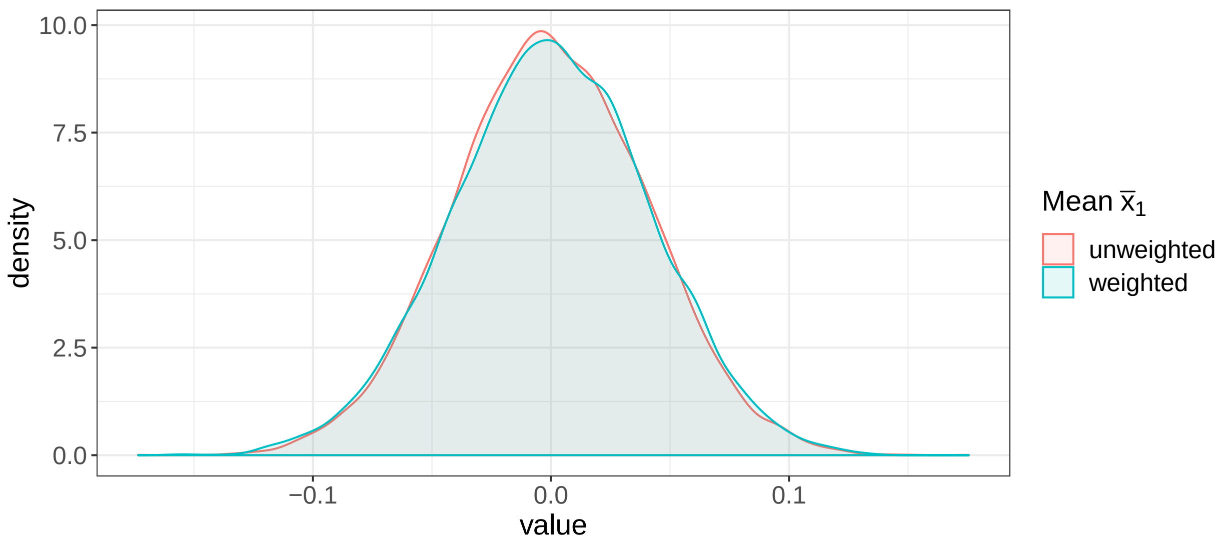

Figure B.1: Comparison of the distributions of the sample means (10^4 samples, sampled from a uniform distribution) for the unweighted and weighted case. The forgetting factor for the weighted estimator is set to $\lambda = 0.99$. The unweighted estimator computes the mean for samples of size $n_{mem} = 199$. Both distributions match closely, confirming our finding regarding the memory of the exponentially decaying estimator.

Intuitively, one might assume for the second question that the sample size n for the ordinary mean corresponds to the value of the normalization factor W_n since for large n, W_n converges – according to

$$W_n = \sum_{i=1}^{n} w'_i = \sum_{i=1}^{n} \lambda^{n-i} = \frac{1-\lambda^n}{1-\lambda}$$

– towards a fixed value (for $\lambda < 1$):

$$\lim_{n\to\infty} W_n = \lim_{n\to\infty} \frac{1-\lambda^n}{1-\lambda} = \frac{1}{1-\lambda}.$$

For example, for a forgetting factor of $\lambda = 0.99$, W_n converges towards $W_n = 100$. Hence, one would be tempted to assume that $n_{mem} = W_n = 100$. However, if we test this hypothesis in a small simulation, we find that memory must be larger. Hence, we have to find another way to estimate the memory n_{mem}. We can do this by actually computing the covariance matrix for the weighted mean. The derivations are shown in the previous section, in which the following resulting equation is found:

$$\begin{aligned}\operatorname{cov}(\bar{X}_n, \bar{Y}_n) &= \sum_{i=1}^{n} w_i^2 \operatorname{cov}(X_i, Y_i) \\ &= \operatorname{cov}(X, Y) \sum_{i=1}^{n} w_i^2.\end{aligned}$$

If we extend the above equation to the setting with exponentially decaying weights, we obtain:

$$\begin{aligned}\sum_{i=1}^{n} w_i^2 = \sum_{i=1}^{n} \Big(\frac{\lambda^{n-i}}{W_n}\Big)^2 = \sum_{i=1}^{n} \frac{(\lambda^{n-i})^2}{W_n^2} &= \sum_{i=1}^{n} \frac{\lambda^{2(n-i)}}{\Big(\sum_{i=1}^{n} \lambda^{n-i}\Big)^2} \\ &= \sum_{i=1}^{n} \frac{\lambda^{2(n-i)}}{\Big(\frac{1-\lambda^n}{1-\lambda}\Big)^2} = \Big(\frac{1-\lambda}{1-\lambda^n}\Big)^2 \sum_{i=1}^{n} \lambda^{2(n-i)} \\ &= \Big(\frac{1-\lambda}{1-\lambda^n}\Big)^2 \frac{1-\lambda^{2n}}{1-\lambda^2}.\end{aligned}$$

For $0 < \lambda < 1$ and large n, the above expression converges towards

$$\lim_{n\to\infty} \sum_{i=1}^{n} w_i^2 = \lim_{n\to\infty} \Big(\frac{1-\lambda}{1-\lambda^n}\Big)^2 \frac{1-\lambda^{2n}}{1-\lambda^2} = \frac{(1-\lambda)^2}{1-\lambda^2}.$$

Due to the central limit theorem (CLT), we know that the ordinary mean vector $\bar{\mathbf{X}}_n$ is distributed as

$$\bar{\mathbf{X}}_n \sim \mathcal{N}\big(\mu_X, \frac{1}{n}\mathbf{\Sigma}\big),$$

so that we have to solve the following correspondence for n_{mem}:

$$\begin{aligned}\mathbf{\Sigma}\frac{1}{n_{mem}} &= \mathbf{\Sigma} \cdot \sum_{i=1}^{n} w_i^2 \\ n_{mem} &= \frac{1}{\sum_{i=1}^{n} w_i^2} = \Big(\frac{1-\lambda^n}{1-\lambda}\Big)^2 \frac{1-\lambda^2}{1-\lambda^{2n}},\end{aligned}$$

For large sample sizes n, n_{mem} converges to:

$$\begin{aligned}\lim_{n\to\infty} n_{mem} &= \frac{1-\lambda^2}{(1-\lambda)^2} = \frac{1-\lambda^2}{(1-\lambda)^2}\cdot\frac{1+\lambda}{1+\lambda}\\ &= \frac{1+\lambda}{1-\lambda}\cdot\frac{1-\lambda^2}{(1+\lambda)(1-\lambda)} = \frac{1+\lambda}{1-\lambda}\cdot\frac{1-\lambda^2}{1-\lambda^2}\\ &= \frac{1+\lambda}{1-\lambda}.\end{aligned}$$

For our previous example with $\lambda = .99$, this would mean that the memory of the estimator is approx. $n_{mem} \approx 199$. The distributions of the sample means for the weighted and unweighted estimator are visualized in Fig. B.1.

In this section, we found that the memory of the exponentially decaying estimator of the sample mean and sample covariance has a memory of approximately:

$$n_{mem} \approx \frac{1+\lambda}{1-\lambda}. \tag{B.92}$$

B.3 Relationship of the Mahalanobis Distance and the Chi-Square Distribution

In practice, sometimes (multivariate) Gaussian distributions are used for anomaly detection tasks (assuming that the considered data is approximately normally distributed): the parameters of the Gaussian can be estimated using maximum likelihood estimation (MLE) where the maximum likelihood estimate is the sample mean and sample covariance matrix. After estimating the distribution parameters, one has to specify a critical value (anomaly threshold) that separates the normal data from the anomalous data. One possibility is, to determine the critical value based on the probability density function (PDF). A new data point can then be classified as anomalous if the value of the PDF for this new point is below the critical value. In the univariate case, the boundary separates the lower and upper tails of the Gaussian from its center (mean). For a 2-dimensional distribution, the boundary is an ellipse around the center, and in higher dimensions, the boundary can be described by an ellipsoid. All points on the surface of such an ellipsoid have the same Mahalanobis distance to the center of the distribution. Hence, one can also specify a Mahalanobis distance as an anomaly threshold and separate normal and anomalous data points according to this distance metric. Generally, the Mahalanobis distance is parameter-free and does not require any particular assumptions about the data distribution (such as a normal distribution). However, as we will show in the following, for (roughly) Gaussian-distributed data, the squared Mahalanobis distance follows a Chi-square distribution. This relationship can also be verified empirically with a quantile-quantile plot.

B.3.1 Prerequisites

B.3.1.1 Matrix Algebra

Generally, the product of a $n \times \ell$ matrix $\mathbf{A}$ and a $\ell \times p$ matrix $\mathbf{B}$ is defined as:

$$(\mathbf{AB})_{ij} = \sum_{k=1}^{m} \mathbf{A}_{ik}\mathbf{B}_{kj}$$

Then, the multiplication of a matrix $\mathbf{A}$ with its transpose $\mathbf{A}^\mathsf{T}$ can be written as:

$$\begin{aligned}
(\mathbf{A}\mathbf{A}^\mathsf{T})_{ij} &= \sum_{k=1}^{\ell} \mathbf{A}_{ik}\mathbf{A}^\mathsf{T}_{kj} = \sum_{k=1}^{\ell} \mathbf{A}_{ik}\mathbf{A}_{jk}, \\
\mathbf{A}\mathbf{A}^\mathsf{T} &= \sum_{k=1}^{\ell} \mathbf{a}_k\mathbf{a}_k^\mathsf{T},
\end{aligned} \tag{B.93}$$

where $\mathbf{a}_k$ is the kth column vector of matrix $\mathbf{A}$.

B.3.1.2 Eigenvalues and Eigenvectors

When the eigenvectors of $n \times n$ matrix $\mathbf{A}$ are arranged in a squared $n \times n$ matrix $\mathbf{U}$, where the i-th column represents the i-th eigenvector $\mathbf{u}^{(i)}$, we have:

$$\begin{aligned}
\mathbf{AU} &= \mathbf{U\Lambda} \\
\mathbf{A} &= \mathbf{U\Lambda U}^{-1},
\end{aligned} \tag{B.94}$$

commonly referred to as eigenvalue decomposition, where $\mathbf{\Lambda}$ is a diagonal matrix containing the eigenvalues λ_i of the corresponding eigenvectors. If $\mathbf{A}$ is a symmetric matrix, the eigenvectors are orthogonal (orthonormal) and the matrix $\mathbf{U}$ is orthogonal as well (the product with its transpose is the identity matrix). In this case $\mathbf{U}^{-1} = \mathbf{U}^\mathsf{T}$, and equation (B.94) can be written as $\mathbf{A} = \mathbf{U\Lambda U}^\mathsf{T}$. The square root of $\mathbf{A}$ (written here as $\mathbf{A}^{\frac{1}{2}}$) – such that $\mathbf{A}^{\frac{1}{2}}\mathbf{A}^{\frac{1}{2}} = \mathbf{A}$ – can be written as:

$$\mathbf{A}^{\frac{1}{2}} = \mathbf{U\Lambda}^{\frac{1}{2}}\mathbf{U}^\mathsf{T}, \tag{B.95}$$

$$\begin{aligned}
\mathbf{A}^{\frac{1}{2}} \cdot \mathbf{A}^{\frac{1}{2}} &= \mathbf{U\Lambda}^{\frac{1}{2}}\mathbf{U}^\mathsf{T}\mathbf{U\Lambda}^{\frac{1}{2}}\mathbf{U}^\mathsf{T} = \mathbf{U\Lambda}^{\frac{1}{2}}\mathbf{I\Lambda}^{\frac{1}{2}}\mathbf{U}^\mathsf{T} = \mathbf{U\Lambda U}^\mathsf{T} \\
&= \mathbf{A}.
\end{aligned}$$

The eigenvalue decomposition of the inverse of a matrix $\mathbf{A}$ can be computed as follows, using the associative property of the matrix product:

$$\begin{aligned}\mathbf{A}^{-1} &= \left(\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{-1}\right)^{-1} = \left(\mathbf{U}^{-1}\right)^{-1}\boldsymbol{\Lambda}^{-1}\mathbf{U}^{-1} = \mathbf{U}\boldsymbol{\Lambda}^{-1}\mathbf{U}^{-1} \\ &= \mathbf{U}\boldsymbol{\Lambda}^{-1}\mathbf{U}^{\mathsf{T}} \end{aligned} \tag{B.96}$$

Note that $\boldsymbol{\Lambda}^{-1}$ is again a diagonal matrix containing the reciprocal eigenvalues of $\mathbf{A}$.

B.3.1.3 Linear Affine Transform of a Normally Distributed Random Variable

If we apply a linear affine transform to a normal random variable $X \sim \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)$ with a mean vector $\boldsymbol{\mu}_x$ and a covariance matrix $\boldsymbol{\Sigma}_x$, we obtain a new random variable Y:

$$Y = \mathbf{A}X + \mathbf{b}. \tag{B.97}$$

One can compute the new mean $\boldsymbol{\mu}_y$ and covariance matrix $\boldsymbol{\Sigma}_y$ for Y:

$$\begin{aligned}\boldsymbol{\mu}_y &= \mathbb{E}\{Y\} = \mathbb{E}\{\mathbf{A}X + \mathbf{b}\} = \mathbf{A}\mathbb{E}\{\mathbf{X}\} + \mathbf{b} \\ &= \mathbf{A}\boldsymbol{\mu}_x + \mathbf{b}, \end{aligned} \tag{B.98}$$

$$\begin{aligned}\boldsymbol{\Sigma}_y &= \mathbb{E}\{(Y - \boldsymbol{\mu}_y)(Y - \boldsymbol{\mu}_y)^{\mathsf{T}}\} \\ &= \mathbb{E}\{\left[(\mathbf{A}X + \mathbf{b}) - (\mathbf{A}\boldsymbol{\mu}_x + \mathbf{b})\right]\left[(\mathbf{A}X + \mathbf{b}) - (\mathbf{A}\boldsymbol{\mu}_x + \mathbf{b})\right]^{\mathsf{T}}\} \\ &= \mathbb{E}\{\left[\mathbf{A}(X - \boldsymbol{\mu}_x)\right]\left[\mathbf{A}(X - \boldsymbol{\mu}_x)\right]^{\mathsf{T}}\} \\ &= \mathbb{E}\{\mathbf{A}(X - \boldsymbol{\mu}_x)(X - \boldsymbol{\mu}_x)^{\mathsf{T}}\mathbf{A}^{\mathsf{T}}\} \\ &= \mathbf{A}\mathbb{E}\{(X - \boldsymbol{\mu}_x)(X - \boldsymbol{\mu}_x)^{\mathsf{T}}\}\mathbf{A}^{\mathsf{T}} \\ &= \mathbf{A}\boldsymbol{\Sigma}_x\mathbf{A}^{\mathsf{T}} \end{aligned} \tag{B.99}$$

B.3.2 The Squared Mahalanobis Distance follows a Chi-Square Distribution

In this section we prove the conjecture: ”The squared Mahalanobis distance of a Gaussian-distributed random vector $\mathbf{X}$ and the center $\boldsymbol{\mu}$ of this Gaussian distribution follows a Chi-square distribution.”

B.3.2.1 Derivation Based on the Eigenvalue Decomposition

The Mahalanobis distance between two points $\mathbf{x}$ and $\mathbf{y}$ is defined as

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \mathbf{y})}. \tag{B.100}$$

Thus, the squared Mahalanobis distance of a random vector $\mathbf{X}$ and the center $\boldsymbol{\mu}$ of a multivariate Gaussian distribution is defined as:

$$D = d(\mathbf{X}, \boldsymbol{\mu})^2 = (\mathbf{X} - \boldsymbol{\mu})^\intercal \boldsymbol{\Sigma}^{-1} (\mathbf{X} - \boldsymbol{\mu}), \tag{B.101}$$

where $\boldsymbol{\Sigma}$ is a $\ell \times \ell$ covariance matrix and $\boldsymbol{\mu} \in \mathbb{R}^\ell$ is the mean vector. In order to achieve a different representation of D one can first perform an eigenvalue decomposition on $\boldsymbol{\Sigma}^{-1}$ which is (with Eq. (B.96) and assuming orthonormal eigenvectors):

$$\boldsymbol{\Sigma}^{-1} = \mathbf{U}\boldsymbol{\Lambda}^{-1}\mathbf{U}^{-1} = \mathbf{U}\boldsymbol{\Lambda}^{-1}\mathbf{U}^T \tag{B.102}$$

With Eq. (B.93) we obtain (cf. [15, p. 80]):

$$\boldsymbol{\Sigma}^{-1} = \sum_{k=1}^{\ell} \lambda_k^{-1} \mathbf{u}_k \mathbf{u}_k^\intercal \tag{B.103}$$

where $\mathbf{u}_k$ is the k-th eigenvector of the corresponding eigenvalue λ_k. Plugging (B.103) back into (B.101) results in:

$$\begin{aligned}
D = (\mathbf{X} - \boldsymbol{\mu})^\intercal \boldsymbol{\Sigma}^{-1} (\mathbf{X} - \boldsymbol{\mu}) &= (\mathbf{X} - \boldsymbol{\mu})^\intercal \left(\sum_{k=1}^{\ell} \lambda_k^{-1} \mathbf{u}_k \mathbf{u}_k^\intercal \right) (\mathbf{X} - \boldsymbol{\mu}) \\
&= \sum_{k=1}^{\ell} \lambda_k^{-1} (\mathbf{X} - \boldsymbol{\mu})^\intercal \mathbf{u}_k \mathbf{u}_k^\intercal (\mathbf{X} - \boldsymbol{\mu}) \\
&= \sum_{k=1}^{\ell} \lambda_k^{-1} \left[\mathbf{u}_k^\intercal (\mathbf{X} - \boldsymbol{\mu}) \right]^2 = \sum_{k=1}^{\ell} \left[\lambda_k^{-\frac{1}{2}} \mathbf{u}_k^\intercal (\mathbf{X} - \boldsymbol{\mu}) \right]^2 \\
&= \sum_{k=1}^{\ell} Y_k^2
\end{aligned}$$

where Y_k is a new random variable based on an affine linear transform of the random vector $\mathbf{X}$. According to Eq. (B.98) , we have $\mathbf{Z} = (\mathbf{X} - \boldsymbol{\mu}) \sim N(\mathbf{0}, \Sigma)$. If we set $\mathbf{a}_k^\intercal = \lambda_k^{-\frac{1}{2}} \mathbf{u}_k^\intercal$ then we get $Y_k = \mathbf{a}_k^\intercal \mathbf{Z} = \lambda_k^{-\frac{1}{2}} \mathbf{u}_k^\intercal \mathbf{Z}$. Note that Y_k is now a random variable drawn from a univariate normal distribution $Y_k \sim N(0, \sigma_k^2)$, where, according to (B.99):

$$\begin{aligned}
\sigma_k^2 = \mathbf{a}_k^\intercal \Sigma \mathbf{a}_k &= \lambda_k^{-\frac{1}{2}} \mathbf{u}_k^\intercal \Sigma \lambda_k^{-\frac{1}{2}} \mathbf{u}_k && \text{(B.104)} \\
&= \lambda_k^{-1} \mathbf{u}_k^\intercal \Sigma \mathbf{u}_k && \text{(B.105)}
\end{aligned}$$

If we insert $\mathbf{\Sigma} = \sum_{j=1}^{\ell} \lambda_j \mathbf{u}_j \mathbf{u}_j^\top$ into Eq. (B.105), we get:

$$\begin{aligned}\sigma_k^2 &= \lambda_k^{-1} \mathbf{u}_k^\top \Sigma \mathbf{u}_k = \lambda_k^{-1} \mathbf{u}_k^\top \left(\sum_{j=1}^{\ell} \lambda_j \mathbf{u}_j \mathbf{u}_j^\top \right) \mathbf{u}_k = \sum_{j=1}^{\ell} \lambda_k^{-1} \mathbf{u}_k^\top \lambda_j \mathbf{u}_j \mathbf{u}_j^\top \mathbf{u}_k \\ &= \sum_{j=1}^{\ell} \lambda_k^{-1} \lambda_j \mathbf{u}_k^\top \mathbf{u}_j \mathbf{u}_j^\top \mathbf{u}_k\end{aligned}$$

Since all eigenvectors $\mathbf{u}_i$ are pairwise orthonormal the dotted products $\mathbf{u}_k^\top \mathbf{u}_j$ and $\mathbf{u}_j^\top \mathbf{u}_k$ will be zero for $j \neq k$. Only for the case $j = k$ we get:

$$\begin{aligned}\sigma_k^2 &= \lambda_k^{-1} \lambda_k \mathbf{u}_k^\top \mathbf{u}_k \mathbf{u}_k^\top \mathbf{u}_k = \lambda_k^{-1} \lambda_k ||\mathbf{u}_k||^2 ||\mathbf{u}_k||^2 = \lambda_k^{-1} \lambda_k ||\mathbf{u}_k||^2 ||\mathbf{u}_k||^2 \\ &= 1,\end{aligned}$$

since the the norm $||\mathbf{u}_k||$ of a orthonormal eigenvector is equal to 1. Thus, the squared Mahalanobis distance can be expressed as: $D = \sum_{k=1}^{\ell} Y_k^2$, where $Y_k \sim N(0, 1)$. Now the Chi-square distribution with ℓ degrees of freedom is exactly defined as being the distribution of a variable which is the sum of the squares of ℓ random variables being standard normally distributed. Hence, D is Chi-square distributed with ℓ degrees of freedom.

B.3.2.2 Alternative Derivation Based on the Whitening Property of the Mahalanobis Distance

Since the inverse $\mathbf{\Sigma}^{-1}$ of the covariance matrix $\mathbf{\Sigma}$ is also a symmetric matrix, its square root can be found – based on Eq. (B.95) – to be a symmetric matrix. In this case we can write the squared Mahalanobis distance as

$$\begin{aligned}D &= (\mathbf{X} - \boldsymbol{\mu})^\top \mathbf{\Sigma}^{-1} (\mathbf{X} - \boldsymbol{\mu}) = (\mathbf{X} - \boldsymbol{\mu})^\top \mathbf{\Sigma}^{-\frac{1}{2}} \mathbf{\Sigma}^{-\frac{1}{2}} (\mathbf{X} - \boldsymbol{\mu}) \\ &= \left(\mathbf{\Sigma}^{-\frac{1}{2}} (\mathbf{X} - \boldsymbol{\mu}) \right)^\top \left(\mathbf{\Sigma}^{-\frac{1}{2}} (\mathbf{X} - \boldsymbol{\mu}) \right) = \mathbf{Y}^\top \mathbf{Y} = ||\mathbf{Y}||^2 \\ &= \sum_{k=1}^{\ell} Y_k^2\end{aligned}$$

The multiplication $\mathbf{Y} = \mathbf{W}\mathbf{Z}$, with $\mathbf{W} = \mathbf{\Sigma}^{-\frac{1}{2}}$ and $\mathbf{Z} = \mathbf{X} - \boldsymbol{\mu}$ is typically referred to as a whitening transform, where in this case $\mathbf{W} = \mathbf{\Sigma}^{-\frac{1}{2}}$ is the so called Mahalanobis (or ZCA) whitening matrix. $\mathbf{Y}$ has zero mean, since $(\mathbf{X} - \boldsymbol{\mu}) \sim N(\mathbf{0}, \Sigma)$. Due to the (linear) whitening transform the new covariance matrix $\mathbf{\Sigma}_y$ is the identity matrix $\mathbf{I}$, as shown in the following (using the property in Eq. (B.99)):

$$
\begin{aligned}
\boldsymbol{\Sigma}_y &= \mathbf{W}\boldsymbol{\Sigma}\mathbf{W}^\mathsf{T} = \boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\Sigma}\left(\boldsymbol{\Sigma}^{-\frac{1}{2}}\right)^\mathsf{T} = \boldsymbol{\Sigma}^{-\frac{1}{2}}\left(\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\Sigma}^{\frac{1}{2}}\right)\left(\boldsymbol{\Sigma}^{-\frac{1}{2}}\right)^\mathsf{T} \\
&= \boldsymbol{\Sigma}^{-\frac{1}{2}}\left(\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\Sigma}^{\frac{1}{2}}\right)\boldsymbol{\Sigma}^{-\frac{1}{2}} = \left(\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\Sigma}^{\frac{1}{2}}\right)\left(\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\Sigma}^{-\frac{1}{2}}\right) \\
&= \mathbf{I}.
\end{aligned}
$$

Hence, all elements Y_k in the random vector $\mathbf{Y}$ are random variables drawn from independent normal distributions $Y_k \sim N(0, 1)$, which leads us to the same conclusion as before, that D is Chi-square distributed with ℓ degrees of freedom.