LINK ADAPTATION FOR BICM-OFDM THROUGH ADAPTIVE KERNEL REGRESSION

Sander Wahls and H. Vincent Poor

Department of Electrical Engineering, Princeton University, Princeton, NJ 08544, USA

ABSTRACT

The packet error rate (PER) of wireless BICM-OFDM systems is notoriously difficult to predict analytically. This remains true even if all subcarriers use a common modulation and coding scheme (MCS). Link adaptation, which here shall be understood as the process of adapting the MCS in order to maximize goodput, therefore remains a major challenge. Non-parametric learning is an elegant way to evade the lack of robust analytical models. Learning from multidimensional features is particularly interesting because one-dimensional features can characterize frequency-selective channels only roughly. However, most of the literature discusses methods that are not truly online. Either the computational costs become unbearable over time or the method saturates and effectively stops learning. The modified k nearest neighbors algorithm (k-NN) seems to be the only exception currently. However, k-NN has well-known weaknesses in learning from small sample sets. Two adaptive kernel regression (AKR) methods are therefore proposed instead. Simulation results are reported for a setup in which several practically relevant conditions that have been mostly ignored in previous studies using multidimensional features (imperfect channel knowledge, Doppler shift, feedback delay, collisions) are modeled.

***Index Terms*—** Wireless communication, OFDM, Link adaptation, Machine learning algorithms, Unsupervised learning

I. INTRODUCTION

The transmission of data over wireless channels often is complicated by the high volatility of the medium. It is well known that an adaptation of system parameters such as the modulation and coding scheme (MCS) or transmit power to the current channel conditions can be highly beneficial [1]. Most modern wireless standards such as the IEEE 802.11 standards or 3GPP LTE therefore support some kind of link adaptation. Both standards describe orthogonal frequency division multiplexing (OFDM) systems that employ bit-interleaved coded modulation (BICM). The IEEE 802.11 standards specify the use of a common MCS for all subcarriers. Link adaptation for 802.11-like systems is traditionally based on evaluation of the packet errors and/or simple one-dimensional features such as average SNR [2]. The autorate fallback (ARF) algorithm [3] as well as receiver-based autorate (RBAR) [4] are classic examples. More recent developments are mostly concerned with loss differentiation (the MCS should not change if a packet error was due to a collision with another device) [5], [6], [7], [8] and context awareness (e.g., the link adaptation strategy in a fast moving vehicle should be different from that while sitting on a desk) [9], [10], [11], [12]. There has also been some interest in *online learning strategies* where the MCS is selected using estimates of the packet error rates

This work was supported in part by the German Research Foundation (DFG) under Grant WA 3139/1-1.

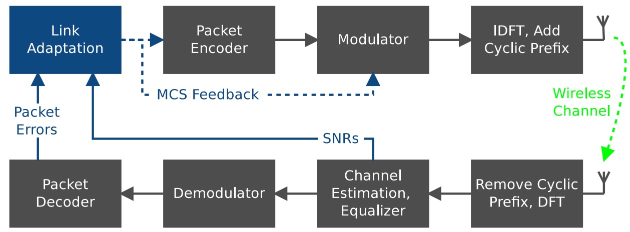

Fig. 1. Link Adaptation for BICM-OFDM

(PERs) [13], [14], [15], [16]. Neither explicit loss differentiation nor explicit context awareness are necessary with this type of algorithm because it can adapt to changing environments. Another trend is to use *multidimensional features* beyond average SNR in order to obtain better predictors for the occurrence of packet errors [17], [18], [19], [20], [21]. Link adaptation strategies based on multidimensional features can be made context aware by suitable extension of the feature vector [11], [12], but most current strategies do not learn online. Thus, they cannot adapt to changes in the environment that are not reflected in the feature vector such as, e.g., collisions or interference from other standards [22]. The modified k nearest neighbor algorithm [23], [24] (k-NN) seems to be the only exception in achieving both benefits (online learning, multidimensional features) simultaneously. However, k-NN has some weaknesses in terms of complexity and performance. (Details follow in Sec. V.) Therefore, in this paper, two new online link adaptation algorithms for multidimensional features are proposed.

II. SYSTEM MODEL

In this section, we discuss the simplified BICM-OFDM system depicted in Fig. 1. It is loosely inspired by the IEEE 802.11 standards. The intention is to give the reader an example to keep in mind while discussing the algorithms, but please note that the results in this paper also apply to many modifications of this setup. Fig. 1 will also be the starting point for the simulations. Basic knowledge about OFDM systems is assumed from henceforth [25].

Transmitter: The packet encoder performs the following actions:

1) Receive N_b bits from a data source.
2) Append a cyclic redundancy checksum (CRC).
3) Encode those bits with a convolutional encoder.
4) Probably puncture the coded bits to increase the code rate.
5) Add an individually encoded header that contains the MCS.
6) Randomly interleave the result.

The packet encoder can be reconfigured on a packet-to-packet basis by specification of an MCS $m \in \{1, 2, \ldots, N_{\mathrm{MCS}}\}$. The MCSs should be ordered by reliability, the first one being the most reliable. The packet is then passed to the modulator, which maps the bits of the packet into complex symbols as specified by some modulation alphabet. Again, the modulator can be reconfigured from packet to packet by specification of an MCS. The number N_b of uncoded bits per packet is always chosen such that the number of resulting complex symbols is equal to some fixed constant N_s. The complex symbols are passed through an inverse discrete Fourier transform (IDFT) right after a cyclic prefix has been added. The output of one run of the IDFT is called an OFDM symbol. The outputs of the IDFT are then transmitted though the wireless channel.

Receiver: The receiver mainly reverses the actions taken at the transmitter. There are some differences though. An additional block that estimates the channel and performs zero-forcing channel equalization is employed. This block is also assumed to estimate the SNRs (per subcarrier). The packet decoder reverts the actions of the encoder and additionally performs a CRC check to see if the packet has been decoded correctly. The packet errors together with the SNRs are passed to the link adaptation, based on which the next MCS is chosen. The MCS is then fed back to the transmitter.

III. PROBLEM FORMULATION

The purpose of the link adaptation block in Fig. 1 is to select an MCS for each packet such that some criterion is optimized. The literature shows many different criteria for link adaptation. Let $t = 0, 1, 2, \dots$ denote the number of already transmitted packets. In this paper, the goal is to find MCSs $m^\star(t)$ such that the goodputs

$$\mathrm{G}(m,t) := (1 - \mathrm{PER}(m,t)) \times \mathrm{T}(m)$$

are maximized. Here, $\mathrm{PER}(m,t)$ denotes the instantaneous PER (i.e., the PER that would result if we fix the channel and transmit using only the MCS m), and $\mathrm{T}(m) =$ code rate$\times$bits per symbol$\times$ symbols per second denotes the throughput of the MCS m. The following MCS selection rule returns an optimal MCS:

$$m^\star(t) := \max\left\{ m : \mathrm{G}(m,t) = \max_{\tilde{m}\in\{1,\dots,N_{\mathrm{MCS}}\}} \mathrm{G}(\tilde{m},t) \right\}. \quad (1)$$

Although the instantaneous PER is not available in a real system, it is common to evaluate the MCS selection rule (1) using predictions of the instantaneous PER in order to choose the MCS. The problem discussed in the remainder of this paper therefore is the prediction of the instantaneous PERs. Special attention will be paid to the following two practical constraints:

1) The computational and memory requirements of the prediction algorithm should be bounded independently of the time.
2) The instantaneous PER can be influenced by external factors. Thus, the prediction algorithm should react to changes in the distribution of the packet errors in a timely manner.

Remark 1. Any prediction algorithm that satisfies the second constraint in particular does not require a-priori training.

Remark 2. The maximum in the MCS selection rule (1) lets the link adaptation prefer higher MCSs. This supports a fast startup.

Remark 3. Other popular link adaptation formulations reduce to the PER prediction problem as well. For example, the MCS selection rule used in [17] is $m^\dagger(t) = \max\left(\{1\} \cup \{m : \mathrm{PER}(m,t) \le P\}\right)$.

IV. ADAPTIVE KERNEL REGRESSION

Consider two random variables, $\mathbf{X}$ from $\mathbb{R}^d$ and Y from $\mathbb{R}$, and a set $\mathcal{H}$ of square integrable functions mapping $\mathbb{R}^d$ to $\mathbb{R}$. Then, the regression problem is to find an $f^\star \in \mathcal{H}$ such that

$$\mathbb{E}\left[(Y - f^\star(\mathbf{X}))^2\right] \le \mathbb{E}\left[(Y - f(\mathbf{X}))^2\right] \qquad \forall f \in \mathcal{H}. \quad (2)$$

In link adaptation, $\mathbf{X}$ represents the channel state and $Y \in \{0,1\}$ indicates packet errors with one being an error. Consequently, the instantaneous PER can be expressed as the a posteriori probability

$$\eta(\mathbf{x}) = \mathbb{P}\left[Y = 1 | \mathbf{X} = \mathbf{x}\right]. \quad (3)$$

The relation to regression is as follows. It is known that $f^\star = \eta$ satisfies (2) for *any* square-integrable f, even if not from $\mathcal{H}$, since $Y \in \{0,1\}$ [26, Ch. 2.1]. Furthermore, if $\mathcal{H}$ is a convex set of continuous functions, then there is exactly one solution to (2) in $\mathcal{H}$. This solution is an optimal approximation of η in the least squares sense [27, Lem. 5]. Hence, for suitable $\mathcal{H}$, the instantaneous PER may be approximated by solving a regression problem.

IV-A. Basic Idea of Kernel Regression

In kernel regression, a weighted superposition of localized kernel functions is used to approximate the optimal regression function $f^\star$ from finitely many observations $\{\mathbf{x}(i), y(i)\}_{i=1}^N$ of $\mathbf{X}$ and Y:

$$\hat{f}^\star(\mathbf{x}) := \sum_{i=1}^{N} \alpha_i(\mathbf{x}) K(\mathbf{x}, \mathbf{x}(i)).$$

Here, $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a kernel, and the $\alpha_i : \mathbb{R}^d \to \mathbb{R}$ are weighting functions. The Nadaraya-Watson estimator is a classic example of a kernel regression method [28], [29]. It corresponds to the choice $\alpha_i(\mathbf{x}) = y(i)/\sum_{i=1}^N K(\mathbf{x}, \mathbf{x}(i))$ with the Gaussian kernel $K(\mathbf{x}, \mathbf{u}) = \exp(-0.5\|\mathbf{x} - \mathbf{u}\|^2/h^2)$. The parameter h is called the bandwidth of the kernel. It controls the smoothness of the regression function $\hat{f}^\star$.

Remark 4. The space of estimators in kernel regression, $\mathcal{H}$, is the linear hull of the continuous functions $K(\cdot, \mathbf{x}_i)$ and thus convex.

IV-B. Adaptive Kernel Regression

In many applications, $\hat{f}^\star(\mathbf{x})$ has to be evaluated repeatedly while additional observations of $\mathbf{X}$ and Y are being made as time goes by. These additional observations can be used to improve the kernel estimator. Assume an infinite sequence of observations $\{\mathbf{x}(i), y(i)\}_{i=1}^\infty$. In adaptive kernel regression (AKR), the goal is to estimate $f^\star(\mathbf{x}(t))$, $t = 0, 1, 2, \dots$, from the previous observations $\{\mathbf{x}(i), y(i)\}_{i=1}^{t-1}$ and $\mathbf{x}(t)$ using an adaptive estimator of the form

$$\hat{f}^\star(\mathbf{x}, t) := \sum_{i=1}^{N(t)} \alpha_i(\mathbf{x}, t) K(\mathbf{x}, \mathbf{c}_i(t)),$$

where now $\{\alpha_i(\mathbf{x}, t)\}_{i=1}^{N(t)}$ denotes a set of time-varying weights, and $\{\mathbf{c}_i(t)\}_{i=1}^{N(t)}$ denotes a codebook. The size of the codebook should be upper bounded independently of the time, $N(t) \le N_{\max}$, in order to restrict the computational and memory requirements of the algorithm. In this paper, two adaptive strategies are considered.

Nadaraya-Watson with Merging (NWM): This is a stripped down variant of the algorithm in [30]. Initially, while $t \le N_{\max}$, the first $t-1$ observations form the codebook for a common Nadaraya-Watson estimator. However, once $t > N_{\max}$, new observations are merged into the codebook. The algorithm is given by

$$\begin{aligned} i^\star(t) &\in \operatorname*{argmin}_{i\in\{1,\dots,N(t-1)\}} \|\mathbf{c}_i(t-1) - \mathbf{x}(t)\| \text{ (choose any)} \\ N(t) &= \begin{cases} N(t-1) + 1 & \text{if } N(t-1) < N_{\max} \\ N(t-1) & \text{otherwise} \end{cases} \end{aligned}$$

$$
\beta_i(t) = \begin{cases} y(t) & \text{if } i = N(t) \wedge N(t-1) < N_{\max} \\ \delta\beta_{i^\star(t)}(t-1) & \\ +(1-\delta)y(t) & \text{if } i = i^\star(t) \wedge N(t-1) = N_{\max} \\ \beta_i(t-1) & \text{otherwise} \end{cases}
$$

$$
\mathbf{c}_i(t) = \begin{cases} \mathbf{x}(t) & \text{if } i = N(t) \wedge N(t-1) < N_{\max} \\ \delta\mathbf{c}_{i^\star}(t-1) & \\ +(1-\delta)\mathbf{x}(t) & \text{if } i = i^\star(t) \wedge N(t-1) = N_{\max} \\ \mathbf{c}_i(t-1) & \text{otherwise} \end{cases}
$$

$$
\alpha_i(\mathbf{x},t) = \beta_i(t) / \textstyle\sum_{j=1}^{N(t)} K(\mathbf{x}, \mathbf{c}_j(t)).
$$

Here, $\delta \in [0,1]$ is a constant that governs the influence of new observations during the merging process.

Quantized Kernel LMS (QKLMS): This method is a minor modification of the quantized kernel least mean squares (QKLMS) algorithm [31]. The idea is to run the ordinary least mean squares algorithm in an infinite-dimensional reproducing kernel Hilbert space. The quantization in QKLMS can be controlled through a threshold $\epsilon \geq 0$. Any new observation $(\mathbf{x}_i, y_i)$ with $\mathbf{x}_i$ not being ϵ-close in a certain sense is added to a codebook. Otherwise, the weight of the nearest codebook element is updated using y_i. There is no explicit constraint on the codebook size. Hence, we drop the oldest entry if the codebook size reaches $N_{\max} + 1$ in order to enforce one. The resulting update at time t is as follows:

$$
i^\star(t) \in \underset{i \in \{1,\ldots,N(t-1)\}}{\operatorname{argmin}} \|\mathbf{c}_i(t-1) - \mathbf{x}(t)\| \text{ (choose any)}
$$

$$
d(t) = \|\mathbf{c}_{i^\star(t)}(t-1) - \mathbf{x}(t)\|
$$

$$
N(t) = \begin{cases} N(t-1)+1 & \text{if } N(t-1) < N_{\max} \wedge d(t) \geq \epsilon \\ N(t-1) & \text{otherwise} \end{cases}
$$

$$
\mathbf{c}_i(t) = \begin{cases} \mathbf{x}(t) & \text{if } i = N(t) \wedge d(t) \geq \epsilon \\ \mathbf{c}_{i+1}(t-1) & \text{if } i < N(t) \wedge d(t) \geq \epsilon \\ & \wedge N(t) = N_{\max} \\ \mathbf{c}_i(t-1) & \text{otherwise} \end{cases}
$$

$$
e(t) = y(t) - \hat{f}^\star(\mathbf{x}(t), t-1),
$$

$$
\alpha_i(\mathbf{x},t) = \begin{cases} \mu e(t) & \text{if } i = N(t \wedge d(t) \geq \epsilon \\ \alpha_i(\mathbf{x}, t-1) + \mu e(t) & \text{if } i = i^\star(t) \wedge d(t) < \epsilon\,. \\ \alpha_i(\mathbf{x}, t-1) & \text{otherwise} \end{cases}
$$

Here, $\mu > 0$ is a step size. The threshold $\epsilon \geq 0$ allows to control the diversity of the codebook. The Gaussian kernel is used again.

V. PRIOR WORK

The prediction of PERs has received much attention in the literature, but most prior work considers only one-dimensional features (e.g., average SNR). Multidimensional features are better suited to represent frequency-selective channels. The subsampled ordered SNR vector, for instance, has been shown to lead to superior classification ratios when compared to various common one-dimensional features [17]. Inspired by this observation, several proposals have been made for PER prediction based on multidimensional features. The original proposal was to perform *k nearest neighbor regression* (k-NN) on an a priori fixed set of training data [17]. Predictions are made from the average number of packet errors among the k observations in the training set that are closest to the queried feature. This first approach however is ill-suited for practical application because it violates the second constraint discussed in Sec. III. Hence, two proposals for an online k-NN have been made. Both maintain two databases for each MCS, one for features that resulted in packet errors and one for features that did not. The first proposal uses age-based updating, where simply the last $N_{\max}$ observations are being stored [23]. The diversity of the features can be low with this approach (old, but useful features will be forgotten). Density-based updating has been proposed as an alternative [24, Ch. 4.3.4]. If the density ρ of a new feature and its k nearest neighbors is high, then the oldest observation among the neighbors is replaced with the new observation. Otherwise, the oldest among all observations is replaced. Both methods satisfy the constraints in Sec. III. However, there are two shortcomings:

Carrier Frequency	2.4 GHz
Bandwidth	20 MHz
Doppler Shift	$0 \ldots 111.5$ Hz
Sample Time	$4/52$ μs
Length of an OFDM Symbol	52 Samples
Length of Cyclic Prefix	12 Samples
Delay Spread	$0 \ldots 11$ Samples
Number of Taps	$1 \ldots 9$
Signal to Noise Ratio	$5 \ldots 40$ dB
Probability of Collision	$0 \ldots 0.3$
Encoder	Convolutional, Rate $1/2$, Optional Puncturing to $3/4$
Decoder	Viterbi
Packet Length	25 OFDM Symbols

Table I. Simulation Parameters

1) Each neighbor has the same weight in k-NN, even if some neighbors are much closer to the queried feature than others. This is especially relevant if there are only a few observations.
2) The naive implementation of k-NN has a complexity of $O(kN_{\max})$. Even if errors of size $\epsilon > 0$ are tolerated, the complexity bound still is $O((\zeta + k)\log N_{\max})$ for some $\zeta \leq \operatorname{ceil}(1+6/\epsilon)^d$ with d being the feature size [17, Sec. IV-E]. Note that the bound on ζ is huge also for very moderate parameters. For instance, $\operatorname{ceil}(1+6/0.1)^4 = 13{,}845{,}841$.

Both adaptive kernel regression methods proposed in Sec. IV-B resolve these issues. The kernels can be interpreted as a way to weight the observations with respect to their distance to the queried feature. The complexities are only $O(N_{\max})$.

There have been various other proposals for link adaptation based on multidimensional features using kernel regression [20], [32], support vector machines [18], [19], [32], neural networks [21], or decision trees [11]. None of these fulfills the constraints in Sec. III, but note that the kernel regression approach in [20] is actually quite close to the proposed QKLMS method in Sec. IV-B. It also is a constrained version of the kernel least mean squares algorithm, where new observations are added to the codebook only if the angle to the subspace spanned by the codebook is large enough. The problem with the approach in [20] is that existing entries of the codebook (including the weights) can neither be modified nor forgotten. Thus, whenever the environment changes, the codebook has to be expanded in order to learn the new situation. Any hard constraint on the codebook size will therefore eventually cripple the algorithm's ability to adapt to new environments.

MCS	Modulation	Code Rate	MCS	Modulation	C. R.
1	BPSK	1/2	4	16 QAM	1/2
2	QPSK	1/2	5	16 QAM	3/4
3	QPSK	3/4	6	64 QAM	3/4

Table II. Modulation and Coding Schemes

Feature	NWM (h, δ)	QKLMS (μ, h, ϵ)	k-NN (k, ρ)
Sorted SNR	0.5, 0.7	0.2, 2, 1.00	25, 0.5
Mean SNR	2.0, 0.5	0.2, 6, 0.5	25, 5

Table III. Algorithmic Parameters

VI. SIMULATION RESULTS

A simulation environment similar to the setup discussed in Sec. II has been implemented. The source code is available at http://bitbucket.org/wahls/link_adaptation. The simulation parameters have been chosen similar to a typical IEEE 802.11 device. See Tab. I. Note that there is no specific channel model involved. The number of taps, their delays as well as their power profile are chosen uniformly at random. The Doppler shifts are also chosen uniformly at random. Packets are lost due to collisions with a probability also chosen uniformly at random. Thus, the link adaptation has to be able to adapt to a very large variety of channels. *The simulation parameters are changed every* 100 *packets.* The channel itself changes with every sample. No block-fading assumption is being made, but perfect synchronization is assumed. The packet sizes are chosen such that each packet occupies 25 OFDM symbols. The channel frequency response is estimated using pilots. It is used to perform zero-forcing channel equalization as well as link adaptation. The MCSs are listed in Tab. II. The link adaptation derives its features from the channel estimated at the beginning of the *previous* packet. Thus, the feedback is delayed by one packet.

Feature Extraction: Let $\rho_1(t) \leq \rho_2(t) \leq \cdots \leq \rho_{52}(t)$ denote the sorted vector of post-equalization SNRs per subcarrier in dB at time t. Two different feature extraction methods are considered. The first returns the four-dimensional subsampled sorted SNR vector $\begin{bmatrix} \rho_5(t) & \rho_{10}(t) & \rho_{20}(t) & \rho_{40}(t) \end{bmatrix}^T/4$. Except for the logarithmic scale, which keeps the variance of the features low, this is the method considered in [17]. The other method simply returns the mean SNR $\sum_{i=1}^{52} \rho_i(t)/52$.

Exploration Strategy: In order to avoid conservative MCS selection caused by outdated observations, seemingly suboptimal MCSs have to be selected from time to time. Therefore, the MCS is increased to $\min\{N_{\mathrm{MCS}}, m^\star(t)+1\}$ whenever the usual strategy $m^\star(t)$ has resulted in ten successfully decoded packets in a row.

Parameter Tuning: The algorithms were tuned for a codebook size of $N_{\max} = 100$ by exhaustive search over a predetermined set of possible parameters. Test data was obtained by simulating the transmission of $250,000$ packets using a link adaptation that randomly chose the MCSs. The test data has been partitioned into 100 sets. For each set, the algorithms predicted the expected PERs based on the previous observations. The Bayes decision function was used to classify the packets [26, Ch. 2.1]. The average ratio of correct classification among the last 500 packets was used as the optimality criterion. The resulting parameters are given in Tab. III.

Learning Curves: Fig. 2 shows the ratio of successful classifications for another set of test data with parameters as in Tab. III. The average was taken over 100 simulation runs. The curves were smoothed with a moving average comprising of up to 251 packets. The proposed AKR methods learn faster and achieve a better steady state than k-NN. The performance of k-NN deteriorates over time. Apparently, density-based updating did not preserve the diversity of the codebook. The sorted SNR feature results in better classification performance than mean SNR for all algorithms.

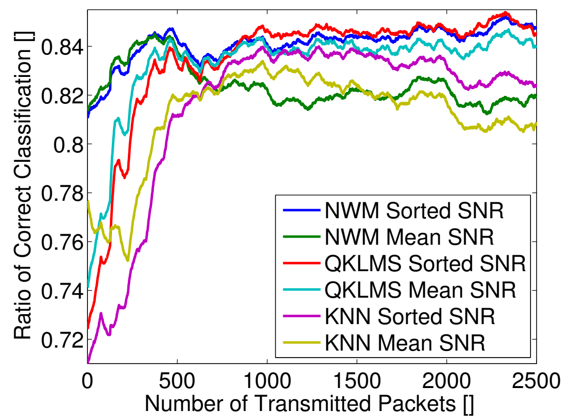

Fig. 2. Learning Curves

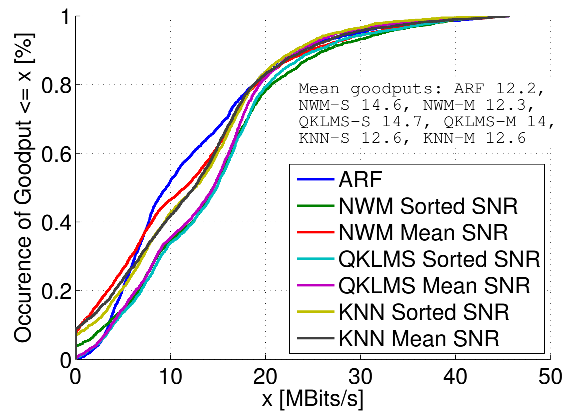

Fig. 3. Cumulative Distribution Function of the Average Goodputs

Goodputs: Fig. 3 shows the link adaptation performance taken over 50 simulation runs of $5,000$ packets each. ARF was included as a baseline algorithm. The AKR methods outperform both ARF and k-NN. They profit from using the sorted SNR feature which k-NN does not (probably because the deterioration gets worse over time). In contrast to QKLMS and ARF, NWM and k-NN fail to adapt in about $5-10\%$ of the channels and give a zero goodput.

VII. CONCLUSION

Two new adaptive kernel regression algorithms have been proposed for PER prediction in BICM-OFDM systems from multidimensional features. The algorithms are suitable for online use. Their computational and memory requirements stay bounded over time. In simulations, the new algorithms outperformed online k-NN in terms of classification accuracy as well as achieved goodput during link adaptation while using less resources. Since NWM failed to adapt to some channels, QKLMS seems to be the preferable method. The benefits of multidimensional features could be confirmed for both proposed methods.

VIII. REFERENCES

[1] A. J. Goldsmith and S.-G. Chua, "Variable-rate variable-power MQAM for fading channels," *IEEE Trans. Commun.*, vol. 45, no. 10, pp. 1218–1230, 1997.

[2] S. Biaz and S. Wu, "Rate adaptation algorithms for IEEE 802.11 networks: A survey and comparison," in *Proc. IEEE ISCC*, Marrakech, Morocco, July 2008.

[3] A. Kamerman and L. Monteban, "WaveLAN®-II: a high-performance wireless LAN for the unlicensed band," *Bell Labs Tech. J.*, vol. 2, no. 3, 1997.

[4] G. Holland, N. Vaidya, and P. Bahl, "A rate-adaptive MAC protocol for multi-hop wireless networks," in *Proc. ACM MobiCom*, Rome, Italy, July 2001.

[5] S. H. Y. Wong, H. Yang, S. Lu, and V. Bharghavan, "Robust rate adaptation for 802.11 wireless networks," in *Proc. ACM MobiCom*, Los Angeles, CA, Sept. 2006.

[6] J. Kim, S. Kim, S. Choi, and D. Qiao, "CARA: Collision-aware rate adaptation for IEEE 802.11 WLANs," in *Proc. IEEE INFOCOM*, Barcelona, Spain, Apr. 2006.

[7] G. Judd, X. Wang, and P. Steenkiste, "Efficient channel-aware rate adaptation in dynamic environments," in *Proc. ACM MobiSys*, Breckenridge, CO, June 2008.

[8] S. Khan, S. A. Mahmuda, K. K. Looa, and H. S. Al-Raweshidy, "A cross layer rate adaptation solution for IEEE 802.11 networks," *Comput. Commun.*, vol. 31, no. 8, pp. 1638–1652, 2008.

[9] P. Shankar, T. Nadeem, J. Rosca, and L. Iftode, "CARS: Context-aware rate selection for vehicular networks," in *Proc. IEEE ICNP*, Orlando, FL, Oct. 2008.

[10] V. Pejovic and E. M. Belding, "A context-aware approach to wireless transmission adaptation," in *Proc. IEEE SECON*, Salt Lake City, UT, June 2011.

[11] J. He, H. Liu, P. Cui, J. Landon, O. Altintas, R. Vuyyuru, D. Rajan, and J. Camp, "Design and experimental evaluation of context-aware link-level adaptation," in *Proc. IEEE INFOCOM*, Orlando, FL, Mar. 2012.

[12] H. Liu, J. He, P. Cui, J. Camp, and D. Rajan, "ASTRA: Application of sequential training to rate adaptation," in *Proc. IEEE SECON*, Seoul, Korea, June 2012.

[13] M. A. Haleem and R. Chandramouli, "Adaptive stochastic iterative rate selection for wireless channels," *IEEE Commun. Lett.*, vol. 8, no. 5, pp. 292–294, 2004.

[14] A. Misra, V. Krishnamurthy, and R. Schober, "Stochastic learning algorithms for adaptive modulation," in *Proc. IEEE ICASSP*, Toulouse, France, May 2006.

[15] T. Joshi, D. Ahuja, D. Singh, and D. P. Agrawal, "SARA: Stochastic automata rate adaptation for IEEE 802.11 networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 11, pp. 1579–1590, Nov. 2008.

[16] J. P. Leite, P. H. P. de Carvalho, and R. D. Vieira, "A flexible framework based on reinforcement learning for adaptive modulation and coding in OFDM wireless systems," in *Proc. IEEE WCNC*, Paris, France, Apr. 2012.

[17] R. C. Daniels, C. M. Caramanis, and R. W. Heath, "Adaptation in convolutionally coded MIMO-OFDM wireless systems through supervised learning and SNR ordering," *IEEE Trans. Veh. Technol.*, vol. 59, no. 1, pp. 114–126, 2010.

[18] S. Yun and C. Caramanis, "Multiclass support vector machines for adaptation in MIMO-OFDM wireless systems," in *Proc. 47th Annual Allerton Conf. Commun. Control Comput.*, Monticello, IL, Sept. 2009.

[19] R. Daniels and R. W. Heath, "Online adaptive modulation and coding with support vector machines," in *Proc. Eur. Wireless Conf.*, Lucca, Italy, Apr. 2010.

[20] S. Yun and C. Caramanis, "Reinforcement learning for link adaptation in MIMO-OFDM wireless systems," in *Proc. IEEE GLOBECOM*, Miami, FL, Dec. 2010.

[21] H. Yigit and A. Kavak, "Adaptation using neural network in frequency selective MIMO-OFDM systems," in *Proc. IEEE ISWPC*, Modena, Italy, May 2010.

[22] N. Golmie, R. E. Van Dyck, A. Soltanian, A. Tonnerre, and O. Rebala, "Interference evaluation of Bluetooth and IEEE 802.11b systems," *Wireless Networks*, vol. 9, no. 3, pp. 201–211, 2003.

[23] R. C. Daniels and R. W. Heath, "An online learning framework for link adaptation in wireless networks," in *Proc. Inf. Theory Appl. Wksp. (ITA)*, San Diego, CA, Feb. 2009.

[24] R. C. Daniels, *Machine Learning for Link Adaptation in Wireless Networks*, Dissertation, UT Austin, Austin, TX, Dec. 2011, http://hdl.handle.net/2152/ETD-UT-2011-12-4509.

[25] H. Bölcskei, "MIMO-OFDM wireless systems: Basics, perspectives, and challenges," *IEEE Wireless Commun.*, vol. 13, pp. 31–37, Aug. 2006.

[26] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*, Springer, New York, 1996.

[27] F. Cucker and S. Smale, "On the mathematical foundations of learning," *Bull. Amer. Math. Soc.*, vol. 39, no. 1, pp. 1–49, 2002.

[28] E. A. Nadaraya, "On estimating regression," *Theory Probab. Appl.*, vol. 9, no. 1, pp. 141–142, 1964.

[29] G. S. Watson, "Smooth regression analysis," *Sankhya Indian J. Stat. Ser. A*, vol. 26, no. 4, pp. 359–372, Dec. 1964.

[30] F. M. Pouzols and A. Lendasse, "Adaptive kernel smoothing regression for spatio-temporal environmental datasets," *Neurocomput.*, vol. 90, pp. 59–65, Aug. 2012.

[31] B. Chen, S. Zhao, P. Zhu, and J. C. Principe, "Quantized kernel least mean square algorithm," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 23, no. 1, 2012.

[32] Y. Ma, "Improving wireless link delivery ratio classification with packet SNR," in *Proc. IEEE Int. Electro/Inf. Technol. Conf. (EIT)*, Lincoln, NE, 2005.