Supervised Learning Approaches to Link Adaptation in Wireless Communication Systems

Anonymous Author(s)

Affiliation
Address
email

Abstract

Current wireless communication systems require link adaptation method to provide consumers with reliable and efficient services. Adaptive modulation and coding (AMC) based on channel state information is the most common way of implementing link adaptation. Traditional implementation of AMC attempts to solve an optimization problem with the goal of maximizing the channel throughput with packet error rate (PER) constraints. Our project here considers the use of machine learning approaches for adaptive modulation and coding. We employ three supervised learning techniques, namely k-nearest neighbor (k-NN), support vector machine (SVM) and random forest (RF), to enable the transmitter to perform AMC based on the knowledge of post-processing SNRs at the receiver (through feedback). Simulation results based on IEEE 802.11n standard show that AMC with machine learning would be promising in real systems.

1 INTRODUCTION

In the information era, traffic in wireless communication systems is increasing rapidly as more and more services have to be provided with reliable and efficient transmissions. The fact is that, current wireless communication systems, especially mobile communication systems, always operate in a dynamic environment where the quality of transmission link keeps changing due to the mobility of transmitter and receiver, the obstacle faced in the communication link and so on. To deal with this, it is better to instill a dynamic strategy into the transmission scheme.

1.1 Adaptive modulation and coding (AMC)

Compared to traditional design approaches, link adaptation technique has proven to be a necessity for improving the system spectral efficiency while maintaining the quality of service (QoS) at the desired level given the dynamic nature of the wireless channel. Such design can be implemented by using adaptive modulation and coding at the physical layer to match time-varying channel conditions. That is to say, we use high-order modulation scheme and larger coding rate when the channel condition is good enough to fully utilize the channel bandwidth, and we use low-order modulation scheme and smaller coding rate to keep the bit error rate (BER) small when the channel condition is bad. By adapting the modulation and coding scheme (MCS) of the transmitted signal according to the instantaneous channel state, we can play with the tradeoff between efficiency and reliability of wireless transmission and optimize the transmission performance.

1.2 Conventional and Current Implementation of AMC

Conventional method of implementing AMC technique is to formulate an optimization problem with some error constraints. By maximizing a certain closed-form objective function representing the transmission rate, we can get the selection thresholds for different modulation and coding schemes. However, such a look-up-table filled with optimal parameters mapping for different channel conditions are not suitable for current wireless communication systems where both
multiple-input-multiple-output (MIMO) technique and orthogonal frequency-division multiplexing (OFDM) technique are used. Then link quality metrics now are not low-dimensional anymore, which makes the look-up-tables method unrealistic. Recent research has shown that classification technique in machine learning can provide a better solution to fast AMC due to its flexibility with high-dimensional link metrics and nonlinearity in the system. We can use machine learning technique to assign compatible modulation and coding schemes to the current channel based on past observations of transmitted data, i.e., training data. Previous work on the implementation of k-NN and SVM in adaptive modulation and coding includes [2] [3] [4]. However, to the best of our knowledge, no systematic empirical research exists on the implementation of random forest in AMC. In this regard, our project considers the implementation of RF in AMC besides the implementation of k-NN and SVM.

The remainder of this paper is organized as follows: Section 2 provides the model for the MIMO-OFDM wireless system. In Section 3, the preparation of data set, including feature extraction and selection, is explained, followed by the introduction on k-NN, SVM and RF. Simulation results based on IEEE 802.11n standard are presented and discussed in Section 4. Conclusion comes last in Section 5 with some suggestions for future research directions.

2 SYSTEM MODEL

Define the number of transmitting antennas as $N_t = 2$, the number of receiving antennas as $N_r = 2$, the number of OFDM symbols per frame as $N_o$, the number of data subcarriers per OFDM symbol $N$, the number of spatial streams $N_s \leq \min\{N_t, N_r\}$.

Given MIMO-OFDM symbol $m \in \{0, 1, \ldots, N_s - 1\}$, subcarrier $n \in \{0, 1, \ldots, N - 1\}$, $H[n] \in \mathbb{C}^{N_r \times N_t}$ as the wireless channel, $F[n] \in \mathbb{C}^{N \times N_s}$ as the linear precoding matrix, $G[n] \in \mathbb{C}^{N_s \times N_t}$ as the linear equalization matrix, and $V[m, n] \in \mathbb{C}^{N_r}$ as the noise at the baseband, then the received signal can be calculated by

$$Y[m, n] = \sqrt{E_s} G[n] H[n] F[n] X[m, n] + G[n] V[m, n]$$

Define $H[n] = G[n] H[n] F[n]$ and $V[m, n] = G[n] V[m, n]$ representing effective quantities to include the combined effects of spatial mapping and spatial equalization. $E_s$ is the expected input signal power if $||F[n]||_2 = 1$. We consider complex Gaussian noise vectors $V[m, n] \sim \mathcal{C}\mathcal{N}(0, \sigma^2 I)$ where each element has variance $\sigma^2$ and mean 0, which results in $\bar{V}[m, n] \sim \mathcal{C}\mathcal{N}(0, \sigma^2 G[n] G^*[n])$.

Here we assume

- $N_s = 1$. We assume we only use one spatial stream in our system for simplicity, which corresponds to beamforming in wireless transmission technique. In this case, we only consider MCS0-MCS7 to be our potential AMC schemes (Table 1).
- Uniform modulation. Every symbol over each spatial stream and every subcarrier is modulated with the same constellation order for each transmitted frame. This corresponds to the mandatory modes supported in IEEE 802.11n.
- Slow fading. The wireless channel is static for all OFDM symbols in a single frame.
- Perfect synchronization. We assume that the transmitter and the receiver are perfectly synchronized in both time and frequency domain.
- Perfect channel knowledge. We ignore the channel estimation inaccuracy at the receiver.
- Zero-Forcing (ZF) linear equalization. For the expression of $G[n]$ in this case, please refer to the equation in Section 3.1.2.

3 METHODS

In this section, I will first explain the preparation of the data set used here, which is very important. It includes building the feature set and MCS labeling for channel realization. Then, the three machine learning approaches for classifications are introduced and discussed.

3.1 Dataset Preparation

3.1.1 raw dataset

Here we use a publicly available data set of $2 \times 2$ MIMO-OFDM system simulations for 8-tap wireless channel using MCS0-MCS15 of IEEE 802.11n (only the first 8 MCSs for single spatial stream

\(^1\)Due to the focus of the paper and the page limits, detailed knowledge on MIMO-OFDM wireless transmission system will not be provided here. People can refer to reference [1] for details.
Table 1: IEEE 802.11N MCS LIST

<table>
<thead>
<tr>
<th>MCS</th>
<th>Ns</th>
<th>M</th>
<th>Code Rate</th>
<th>R_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>i = 0</td>
<td>1</td>
<td>2</td>
<td>1/2</td>
<td>6.5 Mbps</td>
</tr>
<tr>
<td>i = 1</td>
<td>1</td>
<td>4</td>
<td>1/2</td>
<td>13.0 Mbps</td>
</tr>
<tr>
<td>i = 2</td>
<td>1</td>
<td>4</td>
<td>3/4</td>
<td>19.5 Mbps</td>
</tr>
<tr>
<td>i = 3</td>
<td>2</td>
<td>16</td>
<td>1/2</td>
<td>26.0 Mbps</td>
</tr>
<tr>
<td>i = 4</td>
<td>2</td>
<td>16</td>
<td>3/4</td>
<td>39.0 Mbps</td>
</tr>
<tr>
<td>i = 5</td>
<td>3</td>
<td>64</td>
<td>2/3</td>
<td>52.0 Mbps</td>
</tr>
<tr>
<td>i = 6</td>
<td>3</td>
<td>64</td>
<td>3/4</td>
<td>58.5 Mbps</td>
</tr>
<tr>
<td>i = 7</td>
<td>3</td>
<td>64</td>
<td>5/6</td>
<td>65.0 Mbps</td>
</tr>
</tbody>
</table>

Table 2: BEST SUBCARRIER ORDERING SETS FOR k-NN AMC in IEEE 802.11n

<table>
<thead>
<tr>
<th>Ns</th>
<th>d</th>
<th>8 Tap ({n_1, n_2, \cdots, n_d})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>(17)</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>(10, 31)</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>(6, 17, 39)</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>(5, 12, 24, 38)</td>
</tr>
</tbody>
</table>

are considered here). All packets are 128 bytes in length. There are two channel data sets, each
data set contains 32000 channel realizations, so there are overall 64000 channel realizations. Both
the wireless channel impulse response data sets and the corresponding packet error rate simulated
using Monte Carlo simulations are included. The first 1000 channel realization were simulated at
\(SNR = 3\) dB, the second 1000 channels were simulated at \(SNR = 4\) dB, and so on. So the SNRs
of the 32000 channel realization in one channel set range from \(3\) dB to \(34\) dB.

3.1.2 feature extraction and selection

Since the performance of a wireless channel can be completely determined by the channel matrix
and the input SNR, \(\{\{H[n]\}_n=0^N\} E_s/\sigma^2\) can be used to construct the feature space. However, due
to the curse of dimensionality, we prefer to extracting important features out of the feature space to
reduce the computational complexity. Here we use \(d\) post-processing SNRs at the receiver’s end as
the feature set.

\[
F(\text{channel realization}) \triangleq \{\gamma^{n_1}, \gamma^{n_2}, \cdots, \gamma^{n_d}\}
\]

a. calculation of SNR

Common matrices for \(G[n]\) include minimum mean square (MMSE) spatial equalization, where
equalization, where \(G_{ZF}[n] = (F^*[n]H[n]H[n]F[n] + (\sigma^2/E_sI))^{-1}F^*[n]H[n]F[n]\). Here we use ZF receivers,
and the post-processing SNR for \(n_{th}\) subcarrier at the receiver simplifies to

\[
\gamma_{ZF}[n] = E_s/(\sigma^2\|G_{ZF}[n]\|^2_F)
\]

where \(F^*[n]\) and \(H^*[n]\) represent the conjugate transpose matrix of \(F[n]\) and \(H[n]\), respectively, and
\(\|\cdot\|^2_F\) represent the Frobenius norm.

b. subcarrier ordering indexes selection

Here we arrange all the post-processing SNR in increasing order and select the best indexes according
to Table 2 for the SNR as our feature set, which are obtained using a brute-force search over all possible
SNR combinations for the best k-NN performance (optimal for k-NN, not for SVM and RF). The elements in the third column represent the rankings of the SNRs we pick as features after
all the post-processing SNRs have been sorted in increasing order. Define the feature set for the \(q_{th}\)
realization as \(s_q\), namely \(F(q_{th} \text{ channel realization}) = s_q\).

3.1.3 MCS Labeling for Channel Realization

Define the optimal MCS for the \(q_{th}\) realization as \(m_q\). The rule for the selection of the best MCS
scheme for a certain channel realization is that:

\(\text{Due to an IT system reorganization in University of Texas at Austin, the data set is not currently online,}
\text{we personally get the data set from Dr. Daniels. He is trying to upload it recently.}\)
(a) 4-NN AMC that assigns MCS2 to the query point.

(b) Illustration of margins and support vectors in training set feature space for SVM.

Figure 1: a) k-NN  b) SVM

- A MCS scheme is selected only if the respective PER is less than or equal to a certain threshold \( P \). While among those potential MCS schemes that satisfy this constraint, we pick up the MCS scheme that has the highest data rate. So the MCS we pick is

\[
MCS_i = \arg\max_i \{ R_i : PER_i < P \}
\]  

At last, after feature selection and MCS labeling, we get \( \{ s_q \}_{q=1}^W \) and \( \{ m_q \}_{q=1}^W \) as the feature set for the \( W \) realizations and the corresponding MCS class label. Now our data set is ready for processing.

3.2 Machine Learning Techniques

3.2.1 K Nearest Neighbor

The k-nearest neighbor (k-NN) algorithm is one of the most simplest machine learning algorithms, it is a non-parametric classifier that classify an query point by a majority vote from the query point’s neighbors in the feature space, with the query point being assigned to the class most common amongst its k nearest neighbors (k is a positive integer, typically small). If \( k = 1 \), then the query point is simply assigned to the class of its nearest neighbor. While as \( k \) increases, k-NN yields smoother predictions, since we average over more data points.

Algorithm 1 K-NN for AMC [2]

1. Initialize \( n_i = 0 \) \( \forall i \in \{0, 1, 2, 3, 4, 5, 6, 7\} \)
2. for \( a \leftarrow 1 \) to \( k \)
3. \( w_a \leftarrow \arg\min_w \{ d(z_w, q): w \notin \{ w_1, \cdots, w_{a-1}\} \} \)
4. \( n_{m(w_a)} \leftarrow n_{m(w_a)} + 1 \)
5. optionset = \( \arg\max_i \{ n_i \} \)
6. return \( \arg\min_i \{ R_i: i \in \text{optionset} \} \)

For k-NN algorithm, we need a rule to break the tie if two different classes have the same number of neighbors around the query point. If a tie happens, we assign the MCS scheme that has a lower data rate to the query point. This tie-broken rule can ensure the reliability of system since MCS scheme with lower data rate has a lower packet error rate for the same input SNR.

3.2.2 Support Vector Machine

Support Vector Machine (SVM) is one of the best supervised learning algorithms. According to the different forms of the optimization problem, SVM models can be classified into distinct groups. Here we are use the so-called C-SVM which is most commonly used [5].

a. one-against-one classification:

For a training data set \( \{ x_k, y_k \}_{k=1}^W \) of size \( W \), where \( x_k \in \mathbb{R}^d \) is the attribute of the \( k \)-th input data, \( y_k \in \{-1, +1\} \) represents the class of the \( k \)-th data. We would like to find a vector \( w \) and constant \( b \) that satisfy [6]:

\[
\begin{align*}
    w^T \phi(x_k) + b &\geq 1 \quad \text{if } y_k = 1 \\
    w^T \phi(x_k) + b &\leq -1 \quad \text{if } y_k = -1
\end{align*}
\]  

(5)
which can be simplified into
\[ y_k(w^T \phi(x_k) + b) \geq 1, \; \forall k \in \{1, 2, \cdots, W\} \] (6)
where \( \phi(\bullet) \) is a nonlinear function that maps the input attributes into a high-dimensional space. We want to solve the optimization problem
\[
\begin{align*}
\max_{w, b, \gamma} \quad & \gamma \\
\text{s.t.:} \quad & y_k(w^T \phi(x_k) + b) \geq \gamma, \; k = 1, 2, \cdots, W \\
& ||w|| = 1
\end{align*}
\] (7)
To make the algorithm less sensitive to outliers, we add a regularization term to make violations possible, which works for non-linearly separable dataset. After some simplification procedure, the optimization problem becomes:
\[
\begin{align*}
\min_{w, b, \xi} \quad & \frac{1}{2}w^T w + C \sum_{k=1}^{W} \xi_k \\
\text{s.t.:} \quad & y_k(w^T \phi(x_k) + b) \geq 1 - \xi_k \text{ if } x_k \text{ in the ith class,} \\
& \xi_k \geq 0, k = 1, 2, \cdots, W 
\end{align*}
\] (8)
\( \xi_k \) is the amount of violation for the \( kth \) data point. \( C \) is the penalty for data point that violate the rules, and \( C \) must be tuned to optimize classifier performance.
We define kernel function as \( \kappa(x_a, x_b) = \phi(x_a)^T \phi(x_b) \), which is a metric that measures the correlation of feature vectors in the feature space. Usually we use the following kernels \( \kappa(x_a, x_b) \): 1) Linear Kernel \( \kappa(x_a, x_b) = x_a^T x_b \), 2) Polynomial Kernel of degree \( d \) \( \kappa(x_a, x_b) = (x_a^T x_b + 1)^d \).
3) Radial Basis Function (RBF) Kernel \( \kappa(x_a, x_b) = e^{-\lambda ||x_a - x_b||^2} \). In this project, we will only use RBF kernel for C-SVM. \( \lambda \) is a RBF kernel parameter we will tune later.

b. multi-class classification:
When using SVM to solve a multi-class classification problem, usually we divide the original problem into multiple "one-against-one" binary classification problems. If \( h \) is the number of classes, then \( \frac{h(h-1)}{2} \) “one-against-one” classifiers should be constructed and each one deals with data from two different classes. In the end, we use a voting strategy to make the final decision, and a testing data point will be assign to a class with the maximum number of votes.

**Algorithm 2 SVM for AMC**

1. Initialize \( n_i = 0 \) \( \forall i \in \{0, 1, \cdots, 7\} \). \( S = \{(i, j), i, j = \{1, \cdots, h\}, i \neq j\} \)
2. for \((i, j) \in S\) 
3. Solve problem (8) for the two classes in \((i, j)\)
4. Assign class \( m_{(i,j)}(q) \) to query point \( q \)
5. \( n_{m_{(i,j)}(q)} \leftarrow n_{m_{(i,j)}(q)} + 1 \)
6. return \( \arg\max_i \{n_i\} \)

Where \( m_{(i,j)}(q) \) represents the class we assigned to the query point \( q \) according the "one-against-one" classifier we built for class \( i \) and class \( j \) by solving optimization problem (8).

3.2.3 Random Forest
Here we are going to use random forest to assign the MCS labels to different channel realizations. Random Forest is a combination of multiple decision tree predictors that use an ensemble method for classification or regression. It includes the idea of "bagging" and the random selection of features. Since random forest technique has been taught in class, the details of random forest will not be covered here, and reference [10] provides a thorough introduction on random forest.

4 SIMULATION RESULTS AND DISCUSSION
In this section, we employ the three machine learning algorithms to the physical layer of an IEEE 802.11n system. Table 3 provides the system parameters for simulation. Assumptions for the system have been mentioned back in Section 2.
Table 3: Simulation parameters for AMC in IEEE 802.11N

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Selective Channel</td>
<td>8 taps, each tap ∈ C^{NR × NT}</td>
</tr>
<tr>
<td>SNR (dB)</td>
<td>10 log_{10}(E_s/\sigma^2) ∈ {3, 4, ..., 34}</td>
</tr>
<tr>
<td>MCS Classes (i)</td>
<td>i ∈ {0, 1, ..., 7}</td>
</tr>
<tr>
<td>PER Target (P)</td>
<td>0.1 (10% PER )</td>
</tr>
<tr>
<td>Realizations (W)</td>
<td>1000 × 32 = 3.2 × 10^4</td>
</tr>
<tr>
<td>N_r × N_t</td>
<td>2 × 2</td>
</tr>
<tr>
<td>N_s</td>
<td>1</td>
</tr>
</tbody>
</table>

4.1 simulation on k-NN

From Figure 2, we can observe that for k-nearest-neighbor algorithm, the accuracy of classification increases as k and feature dimension d increase, but the gain obtained by increasing k decreases marginally. The increase of k and d will place burden on the computational complexity, so to optimize the k-NN algorithm, appropriate choice of combination of k and d should be searched in consideration of the tradeoff between high classification accuracy and high computation complexity.

4.2 simulation on SVM

4.2.1 cross-validation of training data

For the SVM classifiers, we need to tune the penalty parameter C and the kernel parameter λ to optimize SVM classifiers. By employing a grid 5-fold cross-validation search on the training data for each combination of the classifier parameters, we obtained cross-validation performance and selected the best combination of C and λ for different feature dimension d (Table 4). In Figure 3, we use contour plot to illustrate the results of cross validation for the training data. Then we can use the best parameters to classify our testing data points.

4.2.2 performance of SVM with tuned parameters

Next, we built a classification model with the best parameters C and λ on the original training set and applied this model to the testing set. The performance of these SVM classifiers using the tuned parameters are listed in Table 4. As we can see, as feature dimension d increases, the accuracy of
prediction increases. Despite having a lower processing complexity and memory demand, SVM can exert a comparable performance with k-NN algorithm, which makes SVM a more preferable choice than k-NN.

4.3 simulation on RF

We use Leo Breiman’s algorithm to construct random forest. Figure 4 shows the relationship between RF classification error and number of trees we build, the more trees we build in the random forest, the smaller classification error is. Since the sum of classification error and classification accuracy is one, we can observe that the accuracy RF achieves is comparable with or even higher than SVM. We should remember that the implement of SVM is rather complicated since we need to find penalty $C$ and kernel parameter $\lambda$ by means of the grid-search method using cross-validation. In contrast, RF is easy to use since it has less parameter to tune while the robustness is better than SVM due its inherent nature.

Figure 4: Relationship between RF classification error and number of trees in the forest (bagging parameter: 0.1)
the order indexes of post-processing SNRs that we use as our features are optimal for k-NN, not for SVM and RF. One way to select the optimal features for RF could be to calculate the variable importance (Figure 5). So the performance of SVM and RF we get in this project can be considered to be the performance lower bound).

![Feature importance in Random Forest for post-processing SNRs in 52 subcarriers](image)

**Figure 5**: Feature importance in Random Forest for post-processing SNRs in 52 subcarriers (corresponding to 52 features)

## 5 CONCLUSION

Machine learning techniques with application in link adaptation of wireless communication systems is a promising area worth digging into. In our project, we implemented k-NN, SVM and our novel extension of RF in AMC and demonstrate passable results in a highly frequency-selective channel. While other researchers [2] have already shown that as the channel get less frequency-selective, the performance of AMC using machine learning techniques get better. So we can say that those approaches are promising for real systems. Finally, one possible future direction could be finding the best SNR index for SVM and RF to achieve the optimal SVM-AMC and RF-AMC in link adaptation.

## REFERENCES


