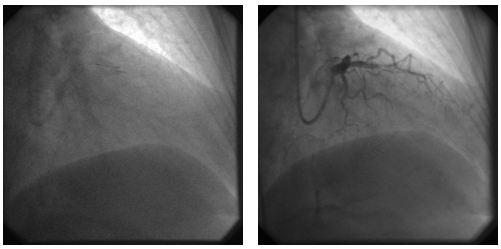
Event Detection using Log-Linear Models for Coronary Contrast Agent Injections

Dierck Matern, Alexandru Paul Condurache and Alfred Mertins
Institute for Signal Processing, University of Luebeck, Ratzeburger Allee 160, Lübeck, Germany
{matern,condura,mertins}@isip.uni-luebeck.de

Keywords: Event Detection, CRF, MEMM, Graphical Modeling

Abstract: In this paper, we discuss a method to detect contrast agent injections during Percutaneous Transluminal Coronary Angioplasty that is performed to treat the coronary arteries disease. During this intervention contrast agent is injected to make the vessels visible under X-rays. We aim to detect the moment the injected contrast agent reaches the coronary vessels. For this purpose, we use an algorithm based on log-linear models that are a generalization of Conditional Random Fields and Maximum Entropy Markov Models. We show that this more generally applicable algorithm performs in this case similar to dedicated methods.



(a) (b)

Figure 1: Two example X-ray images without (a) and with (b) contrast agent.

1 INTRODUCTION

Modern surgical treatment is often image supported. For example, in Percutaneous Transluminal Coronary Angioplasty (PTCA), cardiac X-ray image sequences are used in the treatment of the coronary heart diseases. During the intervention, a contrast agent is injected into the vessels to make them visible and enable a balloon to be advanced to the place of the lesion over a guide wire. Contrast agent is given in occasional bursts, thus, the vessels are not constantly visible. To build a dynamic vessel roadmap (Condurache, 2008) that constantly shows the vessels during the length of the intervention, we need to detect the moment the contrast agent reaches the vessels, that is, the moment they become visible under X-rays. An algorithm to detect this moment is discussed here.

Dedicated methods for this task can be found in (Condurache et al., 2004; Condurache and Mertins, 2009). Here we introduce a more general event detection algorithm based on Log-Linear Models (LLMs) which is closely related to Conditional Random Fields (CRFs) (Wallach, 2004; Lafferty et al., 2001) and Maximum Entropy Markov Models (MEMMs) (McCallum et al., 2000). Our goal is to show that CRF-based event detection, as a general framework, is able to solve event detection problems with accuracy similar to or even better than dedicated methods for the target application.

LLMs, especially CRFs and MEMMs, are finite state models (Gupta and Sarawagi, 2005; Wallach, 2004; Lafferty et al., 2001; McCallum et al., 2000) which are used for segmentation problems (McCallum et al., 2000; Lafferty et al., 2001). For this task, an existing, labeled data set is used to train the model. Then the model is applied to new data sets to generate a corresponding label sequence. For event detection problems, the event is usually unknown, thus we have no labeled data for the events. Therefore, the usual training algorithms for CRFs (Gupta and Sarawagi, 2005) do not apply. An approach when we have partly labeled data (Jiao et al., 2006) is not suitable, because it assumes at least some labels for each class.

MEMMs are Markov models with a maximum entropy approach (McCallum et al., 2000; Gupta and Sarawagi, 2005). From a graphical point of view, they are directed models like Hidden Markov Models (HMMs). If the model is not directed, we have a (linear chain) CRF. This is important for the applications: using a MEMM, the probability to observe a special state depends only on the states before, while

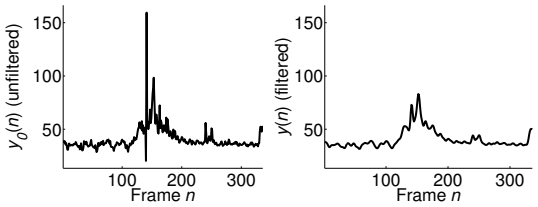


Figure 2: Original vessel feature curve (left hand side) and the filtered histogram feature curves (right hand side).

in CRFs, it depends on the states before and after the actual state. Hence, in MEMMs, we pass forward information, while in CRFs, the information is passed forward and backward. This has two important consequences: (i) the usage of CRFs is more accurate because we use more information for each time step, and (ii) we cannot apply it for online problems, because we have to measure the whole data sequence before the inference. The method we describe here is a trade of of both approaches. We use several frames for the inference for a more accurate decision, while getting only a short and adjustable delay.

We describe a new method to extend the training algorithm such that we can use an LLM for event detection. We call this the Event Detection Log-Linear Model (edLLM). This model gives us the opportunity to conduct the event detection in a more sophisticated manner, respecting that the normal case may consist of several characteristics. Furthermore, we introduce new methods to conduct the inference.

The rest of the paper is structured as follows. In Section 2, we discuss the features we extract for the event detection, the model and the test procedure. In Section 3, we present the results of the experiments. In Section 4, we give a short conclusion.

2 FEATURE EXTRACTION AND MODEL DESCRIPTION

From X-ray images, we extract scalar features that build a feature curve when considering an entire sequence. This is afterwards windowed and transformed into a sequence of feature vectors that we use for event detection. For this purpose, we introduce the edLLMs, and discuss the training and inference.

2.1 Feature Extraction

From each X-ray image we extract a *vessel feature*, sensitive to vessel only. First, a vessel-map showing contrast-enhanced vessels is computed (Condurache and Mertins, 2009). Afterwards we compute the the vessel feature as the 98 percentile of the vessel-map's

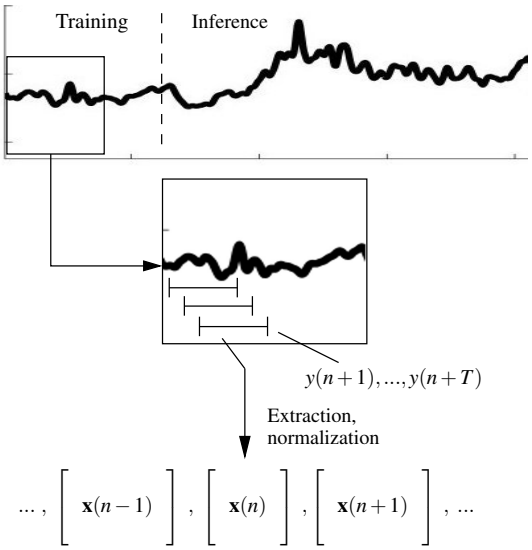


Figure 3: Schematic representation of the feature extraction process. Short sequences of the vessel feature curve are selected, using a sliding window with $T-1$ overlap. From each short sequence $y(n+1), y(n+2) \ldots, y(n+T)$, we extract a normalized feature vector $\mathbf{x}(n)$. This results in a sequence of feature vectors, which we analyze with the edLLM.

gray level histogram. Because this results in a scalar value for each frame, for a sequence of X-ray images this results in a one dimensional sequence of features, that we call a vessel feature curve. Further, to reduce noise, an adaptive filter is applied to this histogram feature curve. This filter is defined as follows (Condurache et al., 2004; Condurache and Mertins, 2009). Let $y_0(n) \in \mathbb{R}$ be the unfiltered vessel feature, then the filtered vessel feature $y(n)$ is

$$y(n) := f(n)y_0(n) + (1-f(n))y(n-1), \quad (1)$$

$$f(n) := \begin{cases} c \in [0,1) & \text{if } y_0(n) - y_0(n-1) \leq 3\hat{\sigma} \\ 1 & \text{otherwise,} \end{cases}$$

where $\hat{\sigma}$ is the estimated standard deviation of the first unfiltered vessel features. This filter smoothes only on stationary intervals. Examples of the unfiltered and filtered curves can be seen in Figure 2. A detailed description how this features are extracted can be found in (Condurache et al., 2004; Condurache and Mertins, 2009).

2.1.1 Feature Vectors

We analyze the data batch-wise. Each batch considers several vessel features corresponding to a specific portion of the feature curve. This portion is selected by means of a sliding window of length T with $T-1$ overlap. Therefore, a batch is given by $y(n), y(n+1), \ldots, y(n+T-1)$, one corresponds to

one frame. For each frame we than compute a feature vector $\mathbf{x}(n)$ that is related to the mean, curvature and slope in each batch. An ordered set of feature vectors is a *feature vector sequence*. A decision is returned for each feature vector and thus for each frame of the video sequence, starting at frame T. In the following, we will discuss the components of the feature vector by introducing the functions used to compute the mean, curvature and slope. These features are general enough to be applicable to many different time series describe local properties of the vessel feature curve. A schematic representation of the feature extraction process can be seen in Figure 3.

We also normalize the feature vectors to reduce the influence of outliers. We call a vessel feature outlier if it is either very large or small in comparison to both, the previous and next vessel feature on the feature curve. The filter in (1) removes most of the relatively smaller outliers, thus we only need to remove very large ones.

2.1.1(a) Mean. The first two components of the feature vector are related to the mean of the feature curve. Let α and β be two scalars with $\alpha, \beta \in (0,1)$. Then $\mu(n+1) := \alpha\mu(n) + (1-\alpha)\sum_{m=0}^{T-1} y(n+m)$ is a slowly adapting mean value, and $\nu(n+1) := \beta\nu(n) + (1-\beta)\mu(n)$ a fast adapting one. We call them "mean values" because for stationary signals, the adapting mean values converge to the mean of the signal with increasing T. α and β are adaptation rates and depend on the frame rate of the analyzed video. Further, let $d_n(m) := y(m) - \mu(n)$ and $\tilde{d}_n(m) := y(m) - \nu(n)$ be differences to those mean values and a vessel feature.

We want to compare the mean value of a batch to the adapting mean values. The first two components of the feature vector are

$$\xi_1(n) := \left(1 - \sqrt{\frac{\sum_{m=0}^{T-1} d_n(m+n)^2}{2 \cdot T \cdot \hat{\sigma}^2}}\right) \cdot 10,$$

$$\xi_2(n) := \left(1 - \sqrt{\frac{\sum_{m=0}^{T-1} \tilde{d}_n(m+n)^2}{5 \cdot T \cdot \hat{\sigma}^2}}\right) \cdot 10,$$

where $\hat{\sigma}^2$ is the estimated variance of the training data. These functions are positive if the mean of the batch is close to the adapting mean values, and negative otherwise. The scaling within the roots control what we call "close". The factor 10 is to emphasize these features, because they act in a manner similar to the statistical test used in (Condurache et al., 2004; Condurache and Mertins, 2009).

2.1.1(b) Curvature. We extract features related to the curvature. These features are computed as the difference of each element of a batch to the mean of a batch. The next T components of the feature vector are:

$$\xi_{2+k+1}(n) := y(n+k) - \frac{1}{T}\sum_{m=0}^{T-1} y(n+m),$$

for $k = 0, 1, 2, \ldots, T-1$.

2.1.1(c) Slope. The last L components of the feature vectors are used to measure the slope of a batch. We define the slope of a batch as the slope of its linear regression, that is $\hat{b}(n) := \arg\min_{b \in \mathbb{R}} \sum_{m=0}^{T-1} (y(n+m) - y(n) - b \cdot m)^2$.

For each batch, we compare these slopes to a set of L sample slopes. Let b_{min} be the minimal slope that is observed in the training batches, and b_{max} the maximal slope. Then a sample slope w_l, $l = 1, 2, \ldots, L$, is defined as $w_l := \frac{L-l}{L-1} b_{min} + \frac{l-1}{L-1} b_{max}$. The function we use to compare slopes of batches to the sample slopes is

$$\xi_{2+T+l}(n) := 1 - \left(\frac{\hat{b}(n) - w_l}{w_{l+1} - w_l}\right)^2$$

for $l = 1, 2, \ldots, L$. This function is positive if a new slope is close to the respective sample slope, and negative else.

2.1.1(d) Outliers. For each batch $y(n), y(n+1), \ldots, y(n+T-1)$, we have defined a vector $\xi(n) := [\xi_i(n)]_{i=1}^{N}$ with $N = 2 + T + L$. The functions $\xi_1, \xi_2, \ldots, \xi_N$ describe properties of each batch, but are affected by outliers. Let $v(n) := \sum_{m=1}^{T-1} (y(n+m) - y(n+m-1))^2$ be the (unnormalized) *local variation*. Then, the normalized outlier-robust feature vector is $\mathbf{x}(n) := \frac{1}{v(n)} \xi(n)$. We call $\mathbf{x}(n)$ the *feature vector* at lag n.

This local variation increases in the presence of large outliers. This normalization does not remove the possibility to detect events in the feature vector sequence, because we assume that any real event is visible for several successive frames.

2.2 Event Detection Log-Linear Model

In the last section, we have described the how to generate the feature vector sequence $\mathbf{x}$. In this section, we describe the edLLM in detail. We first introduce the edLLM in Section 2.2.1 and describe the training and inference afterwards.

2.2.1 Model Description

We propose a LLM that is specialized on event detection. These models process sequences of data in a

manner similar to CRFs and MEMMs. The sequence we use here is the sequence of feature vectors.

Let $\mathbf{x}$ with $\mathbf{x}(n) \in \mathbb{R}^N$ be a feature vector sequence and $\mathbf{s}$ a corresponding state sequence, with $s(n) \in \{\zeta_0, \zeta_1, \zeta_2, \ldots, \zeta_K\}$. We use the convention that the states $\zeta_1, \zeta_2, \ldots, \zeta_K$ describe the normal case, and ζ_0 the event, that is in our case the occurrence of the contrast agent. The training data includes no events: $s(n) \neq \zeta_0$. Using several states for the normal case, we obtain a highly descriptive model that covers the various cases of what is defined to be normal.

For our purpose, we need now to link the training data to the states of the normal case. This is an unsupervised classification problem, as we need to assign each feature vector from the training data one label from the set $\{\zeta_1, \zeta_2, \ldots, \zeta_K\}$. Let y_{min} be the minimum vessel feature in the training data and y_{max} be the maximum one. We define $a_q := \frac{K-q+1}{K} y_{min} + (1 - \frac{K-q+1}{K}) y_{max}$ for $q = 1, 2, \ldots, K+1$, and we label the training data by $s(n) = \zeta_q$ if $a_q \leq \frac{1}{T} \sum_{m=0}^{T-1} y(n+m) < a_{q+1}$.

The idea of the edLLM is to label a feature-vector sequence of a certain length at once. As each feature vector is computed from a batch of vessel features, we use then a larger context for improved decision. We determine the probability of a state sequence, given a sequence of feature vectors, similar to CRFs (Gupta and Sarawagi, 2005). As a difference to CRFs, we work with sequences of feature vectors, that we define with the help of a sliding window. Thus we can use this algorithm online. For the training, however, we use the whole training data in a single sequence of feature vectors. he training is conducted in the same way as for CRFs and MEMMs, except for the penalty term as described in Section 2.2.2.

2.2.1(a) Log-linear model. Let $M \in \mathbb{N}$ be the length of a feature vector sequence $\mathbf{x}_i$ and $t_i \in \mathbb{N}_0$ be a starting index of this sequence, $i = 0, 1, \ldots$ where $i = 0$ denotes the training data. For a shorter notation, let $\mathbf{x}_i := [\mathbf{x}(t_i + m)]_{m=1}^{M}$. We do the same with the states: $s_i(m) := s(t_i + m)$ and $\mathbf{s}_i := [s_i(m)]_{m=1}^{M}$. The probability of $\mathbf{s}_i$, given $\mathbf{x}_i$ and $s_i(0)$ is defined by (Gupta and Sarawagi, 2005)

$$p(\mathbf{s}_i | \mathbf{x}_i, s_i(0)) := \frac{\exp\left(\sum_{m=1}^{M} \lambda^\top \Phi(s_i(m-1), s_i(m), \mathbf{x}_i, m) \right)}{Z(\mathbf{x}_i)}. \quad (2)$$

$Z(\mathbf{x}_i)$ is a normalization value such that $p(\mathbf{s}_i | \mathbf{x}_i, s_i(0))$ is a probability. λ is a weighting vector that identifies features that are important for a successful description of the normal case. The function Φ establishes the relationship between the feature vectors and the states. It is defined by

$$\Phi(s_1, s_2, \mathbf{x}_i, n) := \begin{bmatrix} [\mathbf{x}_i(n) \cdot [\![s_2 = \zeta_k]\!]]_{k=1}^{K} \\ \left[[[\![s_1 = \zeta_j]\!] \cdot [\![s_2 = \zeta_k]\!]]_{j=1}^{K} \right]_{k=1}^{K} \\ \mathbf{x}_i(n) \cdot [\![s_2 = \zeta_0]\!] \end{bmatrix}. \quad (3)$$

with $\mathbf{x}_i(n) \in \mathbb{R}^N$, $\Phi(s_1, s_2, \mathbf{x}_i, n) \in \mathbb{R}^{(N+1)K+K^2}$. $[\![P]\!] = 1$ if the predicate P is true and 0 otherwise (Gupta and Sarawagi, 2005). For each sequence i, we need an initial label $s_i(0) = s(t_i)$. For the training data (that is, $i = 0$), we have no information about $s_0(0) = s(0)$, so we use an arbitrary symbol such that $s(0) \notin \{\zeta_0, \zeta_1, \ldots, \zeta_K\}$ (Gupta and Sarawagi, 2005). For $i = 1$, the initial label is $s_1(0) = s(t_1)$. This is one of the labels used for the training. For $i > 1$, $s_i(0) = s(t_i)$ is determined by the inference at this time step.

2.2.1(b) Example. Assume we have only 2D feature vectors ($N = 2$) and a sequence of length $M = 2$ for some i with $i > 1$, $\mathbf{x}_i(1) = [a, b]^\top$ and $\mathbf{x}_i(2) = [c, d]^\top$. Further, assume we have $K = 2$, that is, $s_i(n) \in \{\zeta_0, \zeta_1, \zeta_2\}$, $n = 1, 2$. Note that with this parameters, $(K+1)^M = 9$ sequences are possible. We assume that the first label is ζ_1, and then we build our example for all possible length 2 sequences.

Using the definitions above, $\Phi(\zeta_1, \zeta_1, \mathbf{x}_i, 1) = [a, b, 0, 0, 0, 0, 0, 0, 0, 0]^\top$, and $\Phi(\zeta_1, \zeta_1, \mathbf{x}_i, 2) = [c, d, 0, 0, 1, 0, 0, 0, 0, 0]^\top$. The fifth entry of $\Phi(\zeta_1, \zeta_1, \mathbf{x}_i, 1)$ indicates the "transition" from state ζ_1 to ζ_1 (the systems stays in state 1). The first two entries are $\mathbf{x}_i(1)$. If we use the state sequence $[\zeta_1, \zeta_2]^\top$, then $\Phi(\zeta_1, \zeta_2, \mathbf{x}_i, 2) = [0, 0, c, d, 0, 1, 0, 0, 0, 0]^\top$, so the third and fourth entry of $\Phi(\zeta_1, \zeta_2, \mathbf{x}_i, 1)$ are $\mathbf{x}_i(2)$. The sixth entry of $\Phi(\zeta_1, \zeta_2, \mathbf{x}_i, 1)$ is 1, due to the transition from ζ_1 to ζ_2. If we assume an event at lag two, that is $s(2) = \zeta_0$, then $\Phi(\zeta_1, \zeta_0, \mathbf{x}_i, 2) = [0, 0, 0, 0, 0, 0, 0, 0, c, d]^\top$.

We further assume that the weighting vector is $\lambda = [\lambda(j)]_{j=1}^{10}] = [1, -1, -1, 1, 1, 0.5, 0.5, 1, 1, 1]^\top$. Then, according to Equation (2), we compute the unnormalized probability as

$$\tilde{p}(\mathbf{s} | \mathbf{x}_i, s(0)) = \exp\left(\sum_{m=1}^{2} \lambda^\top \Phi(s(m-1), s(m), \mathbf{x}_i, m) \right). \quad (4)$$

$s(0)$ is some arbitrary symbol such that $s(0) \notin \{\zeta_0, \zeta_1, \zeta_2\}$. Hence, $\tilde{p}([\zeta_1, \zeta_1]^\top | \mathbf{x}_i, s(0)) = \exp(\lambda^\top \Phi(\zeta_1, \zeta_1, \mathbf{x}_i, s(0))) = \exp(a - b + c - d + 1) =$

$\exp(a-b)\cdot\exp(c-d+1)$. For the second example sequence, $\tilde{p}([\zeta_1,\zeta_2]^\top|\mathbf{x}_i,s(0)) = \exp(a-b+d-c+0.5) = \exp(a-b)\cdot\exp(d-c+0.5)$. The unnormalized probability of an event at lag two is $\tilde{p}([\zeta_1,\zeta_0]^\top|\mathbf{x}_i,s(0)) = \exp(a-b+c+d) = \exp(a-b)\cdot\exp(c+d)$.

Now we want to decide which of these state sequences provides the highest probability. If $c > d-0.25$ and $d < 0.5$, the most probable sequence is $[\zeta_1,\zeta_1]^\top$. If $c < d-0.25$ and $0.25 > c$, it is $[\zeta_1,\zeta_2]^\top$, otherwise, it is $[\zeta_1,\zeta_0]^\top$. In this case, we have observed an event at lag two. Note that these limits depend on the weighting vector λ which includes a scaling.

In this example, we can interpret λ as follows. The first $N=2$ entries of λ describe the state ζ_1, that means, a positive value "prefers" a corresponding positive value in $\mathbf{x}_i$ (if an entry in λ is positive and the corresponding one in $\mathbf{x}_i$ too, the probability for this state is higher than otherwise), a negative in λ a negative one in $\mathbf{x}_i$; a zero in λ can be interpreted as "no influence" on the probability. The next N entries describe the state ζ_2. The next $K^2=4$ entries describe the transition probabilities between the normal case states. The last N entries describe the event.

2.2.2 Parameter Estimation

The weighting vector λ is estimated in the training phase. During training, we maximize $p(\mathbf{s}_0|\mathbf{x}_0,s(0))$ (see Equation (2)) with respect to λ, given the training data and a corresponding state sequence, where $\mathbf{x}_0$ is the training data and $\mathbf{s}_0$ the corresponding label sequence.

Let t_0 be the start index of the training data (i.e., $t_0=0$) and M_0 the number of training feature vectors. Then, we obtain

$$\Phi(\mathbf{s}_0,\mathbf{x}_0) := \sum_{m=1}^{M_0} \Phi(s_0(m-1),s_0(m),\mathbf{x}_0,m).$$

In the training phase, we optimize L_λ, the penalized log-likelihood of $p(\mathbf{s}_0|\mathbf{x}_0,s(0))$ (Gupta and Sarawagi, 2005), defined as

$$\begin{aligned} L_\lambda &:= \log\left(p(\mathbf{s}_0|\mathbf{x}_0,s(0))\right) - F(\lambda) \\ &= \lambda^\top\Phi(\mathbf{s}_0,\mathbf{x}_0) - \log Z(\mathbf{x}_0) - F(\lambda), \quad (5) \end{aligned}$$

where $F(\lambda) := \frac{||\mathbf{D}\lambda||^2}{2} - \tilde{\mathbf{e}}^\top\lambda$ with $\tilde{\mathbf{e}} := [\tilde{e}_k]_{k=1}^{(N+1)K+K^2}$, and

$$\tilde{e}_k := \begin{cases} \sum_{i=1}^{M_0} [\![s(i)\neq\zeta_j]\!] & \text{if } (j-1)\cdot N < k \le j\cdot N, \\ 0 & \text{otherwise.} \end{cases}$$

$F(\lambda)$ is the penalty term. It is necessary for two purposes: to avoid overfitting (Gupta and Sarawagi, 2005), and to adapt the training to our event detection setup. For the former purpose, $\tilde{\mathbf{e}}$ compensates the different numbers of training vectors for each state. For the latter purpose, we define the *penalty matrix* $\mathbf{D}$ by $\mathbf{D} := [D(i,j)]_{i,j=1}^{(N+1)K+K^2}$ with

$$D(i,j) := \begin{cases} 1 & \text{if } i=j,\ i,j \le N\cdot K+K^2 \\ \frac{N}{2N+1} & \text{if } i=j,\ i,j > N\cdot K+K^2 \\ \frac{1}{2N+1} & \text{if } i\neq j,\ i,j > N\cdot K+K^2 \\ \frac{1}{N} & \text{if } i = k\cdot j+N\cdot K+K^2, \\ & \quad k=1,2,\ldots,K, \\ 0 & \text{otherwise.} \end{cases}$$

We use this penalty matrix in contrast to how training is conducted for usual LLMs (Gupta and Sarawagi, 2005; Lafferty et al., 2001) because we assume that we cannot train for ζ_0 in a direct manner, as the training data $\mathbf{x}_0$ does not include any events, that is $s_0(m)\neq\zeta_0$, $m=1,2,\ldots,M_0$.

Because we want to determine λ such that the penalized log-likelihood (see Equation (5)) is maximized, the training of the edLLM is an optimization problem. The gradient of the penalized log-likelihood is

$$\nabla L_\lambda = \Phi(\mathbf{s}_0,\mathbf{x}_0) - E(\mathbf{s}|\mathbf{x}_0,\lambda) - \mathbf{D}\lambda + \tilde{\mathbf{e}},$$

where $E(\mathbf{s}|\mathbf{x}_0,\lambda)$ is the expected value of the sequence $\mathbf{s}$, given $\mathbf{x}_0$ and λ. An efficient algorithm to compute $E(\mathbf{s}|\mathbf{x}_0,\lambda)$ can be found in (Gupta and Sarawagi, 2005). The training processes in several iterations. At iteration τ, λ is updated by $\lambda^{(\tau+1)} := \lambda^{(\tau)} + 0.3\cdot\nabla L_{\lambda^{(\tau)}}$. The constant 0.3 in the update of λ is selected empirically.

2.2.3 Inference

With a trained model, we want to determine the probability of a state sequence for a new feature vector sequence. For the event detection, the most interesting task is to measure the probability to detect an event, that is the probability that the state ζ_0, appears in a feature vector sequence.

As described before, $i>0$ indicate the i-th feature vector sequence of length M that we want to analyze, with t_i as its starting index. We analyze the whole sequence of feature vectors $\mathbf{x}$ batchwise to increase our classification results. These feature vector sequences $\mathbf{x}_i$ do overlap by d vectors, because with $d>0$, we increase the classification results of later feature vector in each feature vector sequence $\mathbf{x}_i$. Because each feature vector corresponds to a frame of the X-ray images, d is actually a delay. We also can set $d=0$ if we cannot accept one, with costs of precision.

For each feature vector sequence $\mathbf{x}_i$, we first want to estimate a state sequence $\mathbf{s}_i$. An exhaustive search

for each possible state sequence corresponding to this feature vector sequence is not feasible, because we would have to test $(K+1)^M$ different state sequences. To circumvent this problem, we define a *forward path* $\alpha_{i,m}$ and a *backward path* $\beta_{i,m}$ (Gupta and Sarawagi, 2005) recursively by

$$\alpha_{i,m}(k) :=$$
$$\frac{1}{Z_\alpha} \sum_{j=0}^{K} \alpha_{i,m-1}(j) \cdot \exp(\lambda^\top \Phi(\zeta_j, \zeta_k, \mathbf{x}_i, m)),$$

and

$$\beta_{i,m}(j) :=$$
$$\frac{1}{Z_\beta} \sum_{k=0}^{K} \beta_{i,m+1}(k) \cdot \exp(\lambda^\top \Phi(\zeta_j, \zeta_k, \mathbf{x}_i, m+1)),$$

where $m = 1, 2, \ldots, M$, Z_α and Z_β are constants such that $\sum_{j=1}^{K} \alpha_{i,m}(j) = \sum_{k=1}^{K} \beta_{i,m}(k) = 1$. $\alpha_{i,m}(k)$ is the probability that we observe ζ_k at lag m in the feature vector sequence $\mathbf{x}_i$, given the previous states, $\beta_{i,m}(j)$ is the probability of state ζ_j at lag m in the feature vector sequence $\mathbf{x}_i$, given the following states. The probability of state ζ_j at time step m in $\mathbf{x}_i$ is given by

$$p(s_i(m) = \zeta_j | \mathbf{x}_i) := \frac{\alpha_{i,m}(j) \cdot \beta_{i,m}(j)}{Z_{\alpha,\beta}}, \qquad (6)$$

where $Z_{\alpha,\beta}$ is a normalization constant such that $p(s_i(m) = \zeta_j | \mathbf{x}_i)$ is a probability.

We use this forward-backward-algorithm to maximize the information we use for labeling, and therefore the reliability of the method. For the same purpose, we also do not apply the inference of the feature vector sequences independently. For the first feature vectors of the sequence $\mathbf{x}_{i+1}$, we pass forward information from $\mathbf{x}_i$ which we have labeled before.

Assume we have calculated $\alpha_{i,m}(k)$ and $\beta_{i,m}(j)$, $j,k = 1, 2, \ldots, K$. To initialize the inference on the next feature vector sequence $\mathbf{x}_{i+1}$, we define the forward path initiation by $\alpha_{i+1,0}(k) := p(s_i(M-d) = \zeta_k | \mathbf{x}_i)$ and the backward path initiation as $\beta_{i+1,M+1}(j) := \frac{1}{K+1}$. Now we can conduct the inference on the feature vector sequence $\mathbf{x}_i$ and use the feature vectors of the sequences before without a new labeling.

In the case of event detection, we are particularly interested in the probability of the event state ζ_0, that is $p(s_i(m) = \zeta_0 | \mathbf{x}_i)$ (see Equation (6)). We can assume that we have detected an event if $p(s_i(m) = \zeta_0 | \mathbf{x}_i) > \theta$ for some $0 < \theta < 1$. However, this method is very sensitive to noise in the probability of the event state, which can occur if the number of training feature vectors is limited or the variation in the data is high. A noise-robust method is discussed next.

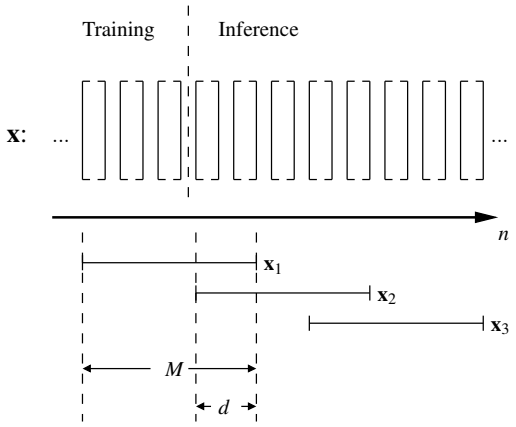


Figure 4: We apply the inference on short sequences of feature vectors, each of length M. These intervals overlap with d feature vectors. Hence, d vectors are re-labeled, using more feature vectors and therefore more information. The first feature vector sequence $\mathbf{x}_1$ is initialized to include several features that belong to the training set.

2.2.3(a) CUSUM test. The probability for the observation of the event state ζ_0 in a feature vector sequence $\mathbf{x}_i$ at lag m is $p(s_i(m) = \zeta_0 | \mathbf{x}_i)$. We need an interpretation of this probability. We use a Cumulated Sum (CUSUM) test (Basseeville and Nikiforov, 1993) to decide if an event has occurred. For this purpose, we define $c(m)$ with

$$c(m) := c(m-1) + p(s_i(m) = \zeta_0 | \mathbf{x}_i) - c_0,$$

where c_0 is estimated such that $c(m) \leq 1$ for $m \leq M_0$, that is, we detect no event in the training data. An event has occurred if $c(m) > 1$. We set $c(m)$ to zero either if an event has occurred or if $c(m) \leq 0$.

3 EXPERIMENTS, RESULTS AND DISCUSSION

We test our algorithm on the same X-ray image sequences used in (Condurache et al., 2004). Some examples are shown in Figure 1, and the vessel feature curves in Figures 2 and 5.

For an event detection algorithm, the data provides several interesting properties that make it challenging and representative for other event detection problems. First, it is generated by real measurements, no simulation. Second, the state of the coronary arteries depends on the heart beat, so a periodicity can be assumed. Without it training could not be applied, because training data does not explain the normal case in the inference. Further, the data depends on machine settings and human interaction, so each sequence can

have very different statistical properties which are, in fact, unknown. Our algorithm is a more general event detection algorithm which can successfully analyze very different time series.

We test the algorithm on nine different sequences of vessel features. To evaluate our results we use the same manual ground-truth proposed in (Condurache et al., 2004). Note that this manual ground-truth is influenced by the quality of the X-ray images, the precise index of the frame where the contrast agent becomes visible is afflicted by small errors. Hence, we rate the experiment as a success if the detection is below 12 frames to the manual ground truth, that is one second.

Because the data is measured in a surgical treatment, the number is limited. To design the features, we randomly have chosen three of the sequences, and the tests are conducted on all nine sequences. As discussed in 2.1, we use a sliding window of length $T = 36$. We produce $N = 41$ dimensional feature vector sequences, with $L = 3$ sample slopes w_l. For each of these feature vectors sequences, we train an edLLM with $K = 5$ on the first $M_0 = 64$ feature vectors. Therefore, the training sequence are the first 99 filtered histogram features (8.25 seconds). We tested different numbers of feature vectors for the inference. With $M = 10$ and a delay $d = 2$, we obtain the best results. Our method is robust with respect to the choice of these parameters.

Our method is robust with respect to the choice of these parameters. We have tested different parameters (L from 3 to 9, T from 10 to 50) without noting any significant change in the obtained results.

Our features are well suited to analyze curves such as the vessel feature curve. Besides mean, curvature and slope, we have also tested other features like skewness or a direct quantization of the features and obtained worse results. The feature vectors that are introduced in Section 2.1.1 are effective for the event detection. Not every component of the feature vectors have the same influence on the results for every sequence, but we could not reduce the feature vectors without decreasing the results for at least one sequence.

In Figure 5, we can see several detection results. Displayed are the manually labeled critical point and the messured one, with both the algorithm proposed in (Condurache et al., 2004) and the new proposal.

In six of the nine experiments, the automatically detected events are below 12 frames (below one second) to the manually ones, in two experiments the events are detected too soon. Those two sequences are contaminated by noise to such an amount that neither the adaptive filter nor our feature extraction can deal with it.

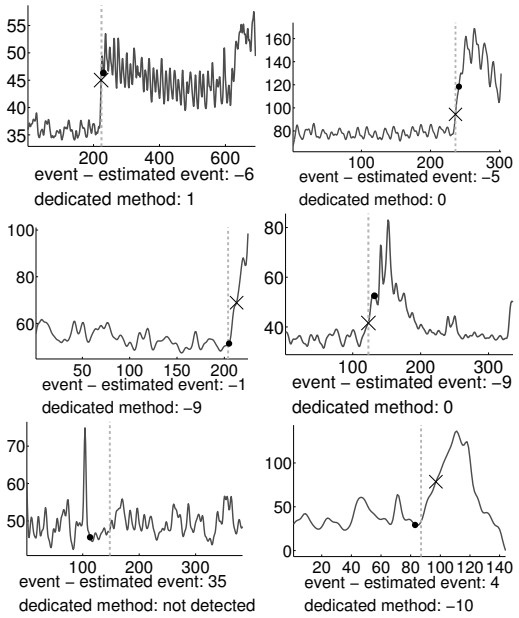


Figure 5: Several sequences of coronary contrast agent injections: the vessel features (dark gray curve), the manually selected critical point (dashed gray line), the detected event (black dot) and the method provided in (Condurache et al., 2004) (black cross) For the last test, the training for the normal case has been reduced to 60 frames because the contrast agent has been injected too early.

4 SUMMARY AND CONCLUSIONS

We have described an event detection method and applied it on the detection coronary contrast agent injections. The results are comparable or even better than the dedicated method (Condurache et al., 2004; Condurache and Mertins, 2009; Condurache, 2008). Due to the decision to analyze batches of frames rather than each frame independently, it is not possible to detect the contrast agent immediately, but after a few frames, however, this lateness is below the human reaction time. Further, we describe a general method able to detect arbitrary events, not just the occurrence of the contrast agent.

The edLLM is more adaptable than the dedicated method. For example, it is not limited to the features we use in Section 2.1. The training implicitly includes a selection of good features, so the set of features can be increased and adapted to more specific or even totally different problems. The features we used in this paper can be interpreted as a starting point: they are general applicable and explain short sequences of

curves. More specialized features can be adapted, the training or inference does not have to be altered.

The delay d we include in our algorithm is used to increase the precision of our labeling and is chosen such that the classification is almost instantaneous. Setting $d = 0$ eliminates the delay, bud decreases the precision with which an event is detected. However, this does not necessary afflict the detection of events itself.

If we compare the edLLM to CRFs and MEMMs from a theoretical point of view, we see a lot of similarities. In fact, we can create a CRF and a MEMM for the event detection by altering the length of the window of feature vectors M and the delay d in the inference. If M is equal to the whole sequence of feature vectors and $d = 0$, we have a CRF, that is, the state sequence can be represented by an undirected graph. If $M = 1$ and $d = 0$, we consider a MEMM. By adapting the functions for the feature vectors and the labeling for the training, we can adapt this method for many other event detection problems, even to offline problems using CRFs and online problems where we can not accept a delay, using MEMMs.

REFERENCES

Basseeville, M. and Nikiforov, I. V. (1993). *Detection of Abrupt Changes: Theory and Application*, page 35ff. Prentice-Hall.

Condurache, A., Aach, T., Eck, K., and Bredno, J. (2004). Fast Detection and Processing of Arbitrary Contrast Agent Injections in Coronary Angiography and Fluoroscopy. In *Bildverarbeitung für die Medizin (Algorithmen, Systeme, Anwendungen)*, pages 5–9.

Condurache, A. P. (2008). *Cardiovascular Biomedical Image Analysis: Methods and Applications*. GCA-Verlag, Waabs, Germany. ISBN 978-3-89863-236-2.

Condurache, A. P. and Mertins, A. (2009). A point-event detection algorithm for the analysis of contrast bolus in fluoroscopic images of the coronary arteries. In *Proc. EUSIPCO 2009*, pages 2337–2341, Glasgow.

Gupta, R. and Sarawagi, S. (2005). Conditional Random Fields. Technical report, KReSIT, IIT Bombay.

Jiao, F., Wang, S., Lee, C.-H., Greiner, R., and Schuurmans, D. (2006). Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 209–216, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lafferty, J. D., Mccallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

McCallum, A., Freitag, D., and Pereira, F. C. N. (2000). Maximum entropy Markov models for information extraction and segmentation. In *ICML*, pages 591–598.

Wallach, H. M. (2004). Conditional Random Fields: An introduction. CIS Technical Report MS-CIS-04-21, University of Pensilvania.