PeakSeg: constrained optimal segmentation and supervised penalty learning for peak detection in count data

Toby Dylan Hocking TOBY.HOCKING@MAIL.MCGILL.CA
Guillem Rigaill RIGAILL@EVRY.INRA.FR
Guillaume Bourque GUIL.BOURQUE@MCGILL.CA
Department of Human Genetics, McGill University, Montréal, Québec, Canada.
Institute of Plant Sciences Paris-Saclay, UMR 9213/UMR1403, CNRS, INRA, Université Paris-Sud, Université d'Evry, Université Paris-Diderot, Sorbonne Paris-Cité, Orsay, France.

Abstract

Peak detection is a central problem in genomic data analysis, and current algorithms for this task are unsupervised and mostly effective for a single data type and pattern (e.g. H3K4me3 data with a sharp peak pattern). We propose PeakSeg, a new constrained maximum likelihood segmentation model for peak detection with an efficient inference algorithm: constrained dynamic programming. We investigate unsupervised and supervised learning of penalties for the critical model selection problem. We show that the supervised method has state-of-the-art peak detection across all data sets in a benchmark that includes both sharp H3K4me3 and broad H3K36me3 patterns.

1. Introduction

1.1. Peak detection in ChIP-seq data

Chromatin immunoprecipitation sequencing (ChIP-seq) is a biological experiment for genome-wide profiling of histone modifications and transcription factor binding sites, with many experimental and computational steps (Bailey et al., 2013). Briefly, each experiment yields a set of sequence reads which are aligned to a reference genome, and then the number of aligned reads are counted at each genomic position (Figure 1). Although these read counts can be interpreted as quantitative data, they are most often interpreted using one of the many available peak detection algorithms (Wilibanks and Facciotti, 2010; Rye et al., 2010; Szalkowski and Schmid, 2011). A peak detection algorithm is a binary classifier for each genomic position. The positive class is enriched (peaks) and the negative class is background noise. Importantly, peaks and background occur in long contiguous segments across the genome.

More concretely, a single ChIP-seq profile on a genomic region with d base pairs can be represented as a vector $\mathbf{y} = \begin{bmatrix} y_1 & \cdots & y_d \end{bmatrix} \in \mathbb{Z}_+^d$ of counts of aligned sequence reads. A peak detection algorithm is a function $c : \mathbb{Z}_+^d \to \{0,1\}^d$ which returns 0 for background noise and 1 for a peak.

1.2. Unsupervised versus supervised peak detection

There are several different kinds of peak patterns that have been observed in different ChIP-seq data sets. For example, Figure 1 shows H3K4me3 data with a sharp peak pattern, and H3K36me3 data with a broad peak pattern. Current state-of-the-art peak detectors from the bioinformatics literature are unsupervised learning algorithms that are designed for specific patterns. For example, the MACS algorithm was initially designed for the sharp pattern in H3K4me3 data (Zhang et al., 2008). Another example is HMCan (Ashoor et al., 2013), whose authors suggest fixed pattern-specific mergeDistance parameters: H3K4me3 $\Rightarrow$ 200, H3K36me3 $\Rightarrow$ 1000. In fact, each unsupervised peak detection algorithm has several numeric parameters, each with defaults that may or may not be optimal for a particular data set. Rather than taking an unsupervised model and default parameters for granted, we propose a new supervised peak detection model that can be efficiently trained to recognize different peak patterns.

In supervised peak detection, there are n labeled samples, and each sample $i \in \{1, \ldots, n\}$ has a profile $\mathbf{y}_i \in \mathbb{Z}_+^d$ and a set of annotated region labels R_i ("noPeaks," "peaks," etc. as in Figures 1 and 2). These labels define a non-convex annotation error function

$$E[c(\mathbf{y}_i), R_i] = \text{FP}[c(\mathbf{y}_i), R_i] + \text{FN}[c(\mathbf{y}_i), R_i] \quad (1)$$

which counts the number of false positive (FP) and false

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

negative (FN) regions, so it takes values in the non-negative integers. The goal of learning is to find a peak caller c with minimal error on some test profiles:

$$\underset{c}{\text{minimize}} \sum_{i \in \text{test}} E[c(\mathbf{y}_i), R_i]. \quad (2)$$

More specifically, we suppose that the training data and the test data exhibit the same pattern type.

1.3. Contributions

The main contribution of this paper is **PeakSeg**, a peak detection model that can be trained using supervised learning, and is effective for several pattern types (sharp and broad peaks). Rather than taking unsupervised model parameters for granted, we propose to train model parameters using labeled data of the same pattern type. Importantly, and in contrast to existing unsupervised approaches which can only be trained using grid search, we propose efficient discrete and convex optimization algorithms for model training. Our method is the first peak detector with an efficient supervised learning algorithm, and the first method that achieves state-of-the-art peak detection across several patterns in the benchmark of Hocking et al. (2014).

A second contribution of this paper is a detailed study of the peak detection accuracy of several unsupervised and supervised penalty function learning methods. The two main results are that the oracle penalty of Cleynen and Lebarbier (2014) is more accurate than asymptotic penalties like the AIC/BIC, and that supervised penalty learning can be used to further increase peak detection accuracy.

2. Related work

Our work is based on and inspired by roughly three types of related work: unsupervised ChIP-seq peak detectors from the bioinformatics literature, maximum likelihood segmentation models, and criteria for model selection.

2.1. Unsupervised ChIP-seq peak detectors

There are literally dozens of unsupervised algorithms for peak detection in ChIP-seq data sets, and the bioinformatics literature contains several published comparison studies (Wilibanks and Facciotti, 2010; Rye et al., 2010; Szalkowski and Schmid, 2011). Hocking et al. (2014) proposed a benchmark of labeled ChIP-seq data sets, with two different histone mark types: H3K4me3 (sharp peak pattern) and H3K36me3 (broad peak pattern). The best peak detection algorithm in these H3K4me3 data was macs (Zhang et al., 2008), and the best for H3K36me3 was HMCan (Ashoor et al., 2013). Both of these algorithms are unsupervised, but were calibrated via grid search using the annotated region labels to choose the best scalar significance threshold hyperparameter. Others have proposed to train the hyperparameters of unsupervised peak detectors (Micsinai et al., 2012; Kumar et al., 2013). In contrast, we propose the supervised **PeakSeg** model which can be efficiently trained using discrete and convex optimization algorithms (Sections 4 and 5).

2.2. Maximum likelihood segmentation models

The **PeakSeg** model we propose in this paper is a constrained version of the Poisson segmentation model of Cleynen et al. (2014). Their unconstrained model can be computed using a dynamic programming algorithm (DPA) (Bellman, 1961), or a pruned dynamic programming algorithm (pDPA) (Rigaill, 2010). Both algorithms are guaranteed to recover the exact solution to the unconstrained model, but there are two important differences. The pDPA is more complicated to implement, but is also computa-

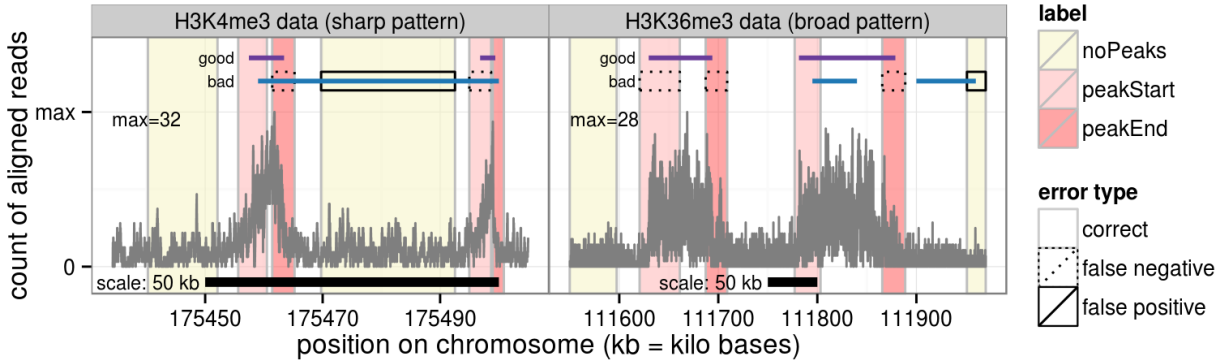

Figure 1. Different ChIP-seq data types have peaks with different patterns. Positions classified as peaks are drawn as line segments for a good peak model with zero errors and a bad peak model with 7 incorrect regions. The goal of supervised peak detection is to learn the data set-specific peak pattern encoded in the annotated region labels (peakStart / peakEnd mean there should be exactly 1 peak start/end somewhere in the region, and noPeaks means there should be no overlapping peaks).

tionally faster than the DPA. For segmenting a sequence of d data points, the pDPA takes on average $O(d \log d)$ time whereas the DPA takes $O(d^2)$ time.

Relative to the unconstrained segmentation model of Cleynen et al. (2014), our proposed **PeakSeg** model has an additional constraint. Rather than searching all possible change-points to find the most likely model with s segments, we propose to constrain the possible change-points to the subset of models that can be interpreted as peaks (with segment means that change up, down, up, etc, see Section 3.2).

Due to the simplicity of its implementation, we propose a constrained dynamic programming algorithm (cDPA) that requires a small modification to the standard DPA solver. Although the DPA is an exact solver for the unconstrained problem, we show that the cDPA is a heuristic that is not guaranteed to solve the constrained problem (Section 4.3). However, we show that it is still very useful and accurate in practice on real data (Section 6).

2.3. Model selection criteria

For each segmentation problem, the proposed cDPA returns a sequence of models with $s \in \{1, 3, \dots, s_{\max}\}$ segments. The model selection problem is choosing a sample-specific number of segments s that maximizes the number of true positive peak detections while minimizing the number of false positives. For example, in Figure 2, an ideal model selection procedure would choose $s \in \{3, 5\}$ segments, since those models have zero errors with respect to the labels (1).

There are many unsupervised penalty functions that select the number of segments without using the labels. For example, the classical AIC (Akaike, 1973) and BIC (Schwarz, 1978) are based on theoretical asymptotic arguments. Several attempts have been made to adapt the BIC for segmentation problems (Yao, 1988; Zhang and Siegmund, 2007), and the non-asymptotic model selection theory of Cleynen and Lebarbier (2014) suggests another penalty function.

In the supervised setting of this paper, there is a training data set of segmentation problems $i \in \{1, \dots, n\}$, with labeled regions R_i that can be used to compute the annotation error (1). Thus, rather than taking a particular unsupervised penalty function for granted, we can use the training data to learn the penalty function with minimum error. In particular in Section 5.3 we propose using the max margin interval regression algorithm of Rigaill et al. (2013), which was originally proposed for learning a penalty function for optimal change-point detection.

3. From unconstrained to constrained maximum likelihoood segmentation

In this section we discuss Poisson maximum likelihood segmentation models for count data. First, we discuss an existing unconstrained model, and then we propose a constraint that makes the model suitable for peak detection.

3.1. Unconstrained maximum likelihood segmentation

For a sequence of d data points $\mathbf{y} \in \mathbb{Z}_+^d$ to segment, we fix a maximum number of segments $s_{\max} \leq d$. The unconstrained maximum likelihood segmentation model is defined as the most likely mean vector $\mathbf{m} \in \mathbb{R}^d$ with $s \in \{1, 2, \dots, s_{\max}\}$ piecewise constant segments:

$$\hat{\mathbf{m}}^s(\mathbf{y}) = \begin{array}{ll} \arg\min\limits_{\mathbf{m} \in \mathbb{R}^d} & \rho(\mathbf{m}, \mathbf{y}) \\ \text{such that} & \text{Segments}(\mathbf{m}) = s, \end{array} \tag{3}$$

where the Poisson loss function is

$$\rho(\mathbf{m}, \mathbf{y}) = \sum_{j=1}^{d} m_j - y_j \log m_j. \tag{4}$$

The model complexity is the number of piecewise constant segments

$$\text{Segments}(\mathbf{m}) = 1 + \sum_{j=2}^{d} I(m_j \neq m_{j-1}), \tag{5}$$

where I is the indicator function.

Although it is a non-convex optimization problem, the sequence of segmentations $\hat{\mathbf{m}}^1(\mathbf{y}), \dots, \hat{\mathbf{m}}^{s_{\max}}(\mathbf{y})$ can be computed in $O(s_{\max} d^2)$ time using dynamic programming (Bellman, 1961), or in $O(s_{\max} d \log d)$ time using pruned dynamic programming (Rigaill, 2010; Cleynen et al., 2014).

We refer to (3) as the "unconstrained" model since $\hat{\mathbf{m}}^s(\mathbf{y})$ is the most likely segmentation of all possible models with s piecewise constant segments ($s-1$ change-points). Several unconstrained models are shown on the left of Figure 2, and for example the second segment of the model with $s = 3$ segments appears to capture the peak in the data. To construct a peak detector c, we first define the peak indicator at base $j \in \{2, \dots, d\}$ as

$$P_j(\mathbf{m}) = \sum_{k=2}^{j} \text{sign}(m_k - m_{k-1}), \tag{6}$$

where $P_1(\mathbf{m}) = 0$ by convention. $P_j(\mathbf{m})$ is the cumulative sum of signs of changes up to point j in the piecewise constant vector $\mathbf{m}$. We define the vector of peak indicators as

$$\mathbf{P}[\mathbf{m}] = \begin{bmatrix} P_1(\mathbf{m}) & \cdots & P_d(\mathbf{m}) \end{bmatrix}. \tag{7}$$

3.2. PeakSeg: constrained maximum likelihood

In general for the unconstrained model $P_j(\mathbf{m}) \in \mathbb{Z}$, which is problematic since we want to use it as a peak detector with binary outputs $P_j(\mathbf{m}) \in \{0, 1\}$.

For example, if $\mathbf{m} = \begin{bmatrix} 1.1 & 1.1 & 2 & 2 & 4 & 4 & 3 \end{bmatrix}$, with two changes up followed by one change down, then $\mathbf{P}(\mathbf{m}) = \begin{bmatrix} 0 & 0 & 1 & 1 & 2 & 2 & 1 \end{bmatrix}$.

Thus we constrain the peak indicator $P_j(\mathbf{m}) \in \{0, 1\}$, which results in the constrained problem

$$\tilde{\mathbf{m}}^s(\mathbf{y}) = \arg\min_{\mathbf{m} \in \mathbb{R}^d} \rho(\mathbf{m}, \mathbf{y}) \qquad \textbf{(PeakSeg)}$$
$$\text{such that } \text{Segments}(\mathbf{m}) = s,$$
$$\forall j \in \{1, \ldots, d\},\ P_j(\mathbf{m}) \in \{0, 1\}.$$

Note that one must specify the number of segments s or, equivalently, the number of peaks $p = (s-1)/2$. Another way to interpret the constrained **PeakSeg** problem is that the sequence of changes in the segment means $\mathbf{m}$ must begin with a positive change and then alternate: up, down, up, down, ... (and not up, up, down). Thus the even-numbered segments may be interpreted as peaks $P_j(\mathbf{m}) = 1$, and the odd-numbered segments may be interpreted as background $P_j(\mathbf{m}) = 0$.

For example, the good peaks in Figure 1 are the second and fourth segments of the **PeakSeg** solution for $s = 5$ segments.

Figure 2 shows a profile where the constraint is necessary for the even-numbered segments to be interpreted as peaks. In particular, it is clear that unconstrained models with $s \in \{5, 7\}$ segments do not satisfy $P_j[\hat{\mathbf{m}}^s(\mathbf{y})] \in \{0, 1\}$ for all positions $j \in \{1, \ldots, d\}$ (since they have up, up, down changes).

4. Algorithms

In this section we first review the existing standard dynamic programming algorithm (DPA) for segmentation, then propose a new constrained dynamic programming algorithm (cDPA). We also give a counter-example which shows that the cDPA is a heuristic for solving the **PeakSeg** optimization problem.

4.1. The standard DPA

The unconstrained model (3) can be computed exactly using a dynamic programming algorithm (DPA). Let t, t' be indices such that $0 \leq t' < t \leq d$, and let the interval $(t', t]$ denote a segment. Then $\sigma_{(t',t]} = \sum_{j=t'+1}^{t} y_j$ is the cumulative sum and $M_{(t',t]} = \sigma_{(t',t]}/(t-t')$ is the mean over the segment $(t', t]$. The optimal Poisson loss (4) for that segment is

$$c_{(t',t]} = \sigma_{(t',t]} \left[1 - \log\left(M_{(t',t]}\right)\right]. \tag{8}$$

Let $\mathcal{L}_{s,t}$ be the optimal Poisson loss of the model with s segments up to data point t, and let $\mathcal{T}_{s,t}$ be the corresponding optimal last change. The key idea behind the DPA is that if we consider an optimal segmentation in s up to t and consider the subsegmentation obtained by discarding the last segment $(\mathcal{T}_{s,t}, t]$ then the resulting segmentation is also optimal in the sense that its loss is equal to $\mathcal{L}_{s-1,\mathcal{T}_{s,t}}$. This can be formally written as a search for the best change-point t'

$$\mathcal{L}_{s,t} = \min_{t'<t} \mathcal{L}_{s-1,t'} + c_{(t',t]}, \tag{9}$$

meaning that the best loss in s segments can be written in terms of the best loss in $s-1$ segments. Using this update rule iteratively for all s from 2 to $s_{\max}$ and for all t from 2 to d we recover the best segmentations in 1 to $s_{\max}$ in $\mathcal{O}(s_{\max} d^2)$ time.

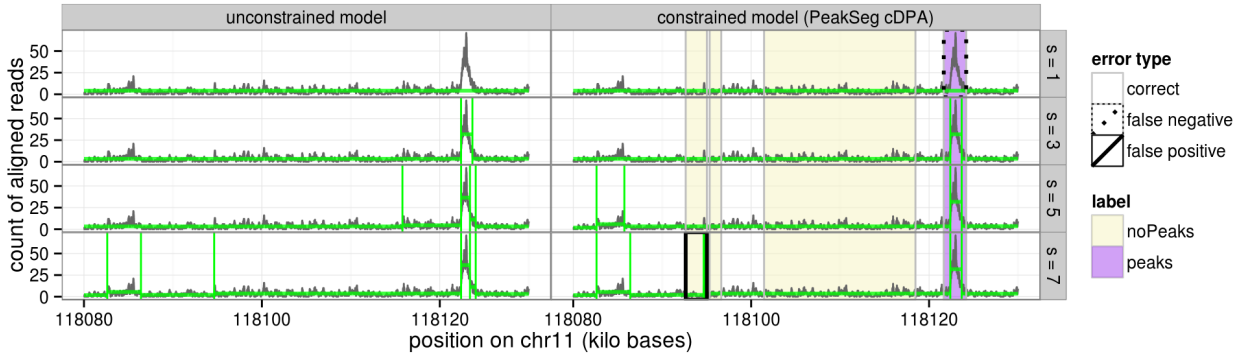

Figure 2. Example profile $\mathbf{y}$ (grey), with green horizontal lines for the segmentation mean $\mathbf{m}$, and green vertical lines to emphasize change-points. For this particular profile $\mathbf{y}$, the unconstrained and contrained models are equivalent $\hat{\mathbf{m}}^s(\mathbf{y}) = \tilde{\mathbf{m}}^s(\mathbf{y})$ for $s \in \{1, 3\}$ segments but not for $s \in \{5, 7\}$. For the constrained models, the even-numbered segments are interpreted as peaks, whose accuracy can be quantified using the annotated region labels (peaks means there should be at least 1 overlapping peak).

4.2. cDPA: a simple modification

In order to force the segmentation to obey the constraints of the **PeakSeg** model, we introduce a simple modification of the previous update rule (9). Up to now we considered all possible change-points t' smaller than the current t. Here we propose to consider only those such that the resulting segmentation can be interpreted as a succession of peaks. To make this idea more precise we introduce the notation $\mathcal{E}_{s,t}$ which is the last empirical mean of the best segmentation in s segments up to t.

Then if s is even (peak segment) the mean of segment $(t', t]$ should be larger than the previous one and in the update rule we only consider t' such that

$$t' < t \mid \mathcal{E}_{s-1,t'} < M_{(t',t]}. \tag{10}$$

On the contrary, if s is odd (background segment) the mean of segment $(t', t]$ should be smaller than the previous one and we consider only t' such that

$$t' < t \mid \mathcal{E}_{s-1,t'} > M_{(t',t]}. \tag{11}$$

In the end we get the following algorithm:[1]

Algorithm 1 Constrained dynamic programming (cDPA)

Require: $\mathbf{y} \in \mathbb{Z}_+^d$, $s_{\max} \in \{2, \dots, d\}$.
For ($s = 2$ to $s_{\max}$) **do** and **For** ($t = s$ to d) **do**
  switch (s)
    case even**:** $T \leftarrow \{t' < t \mid \mathcal{E}_{s-1,t'} < M_{(t',t]}\}$
    case odd**:** $T \leftarrow \{t' < t \mid \mathcal{E}_{s-1,t'} > M_{(t',t]}\}$
  if $T == \emptyset$ **then**
    $\mathcal{L}_{s,t} \leftarrow \infty$
  else
    $\mathcal{T}_{s,t} \leftarrow t^* \leftarrow \arg\min_{t' \in T} \mathcal{L}_{s-1,t'} + c_{(t',t]}$
    $\mathcal{L}_{s,t} \leftarrow \mathcal{L}_{s-1,t^*} + c_{(t^*,t]}$
    $\mathcal{E}_{s,t} \leftarrow M_{(t^*,t]}$
  end if
End for and **End for**
return $s_{\max} \times d$ matrix $\mathcal{T}_{s,t}$ of optimal change points.

Note that we do not need to precompute the $d(d-1)/2$ values of $c_{(t',t]}$ and $M_{(t',t]}$. That is because the optimal cost $c_{(t',t]}$ and empirical mean $M_{(t',t]}$ are simple functions of $\sigma_{(t',t]} = \sum_{j=t'+1}^{t} y_j$ which can be computed in $\mathcal{O}(1)$ given the cumulative sum of the data-points, $\sum_{j=1}^{t} y_j$. Hence the computational requirements of the cDPA are the same as the DPA: $\mathcal{O}(s_{\max} d)$ memory and $\mathcal{O}(s_{\max} d^2)$ time.

Note also that for both the even and odd update the set of possible changes T might very well be empty. This means that no segmentation in s segments to point t satistifying the **PeakSeg** model constraint was found, so we simply set $\mathcal{L}_{s,t} = \infty$.

Hence the cDPA is not guaranteed to recover a segmentation. In particular for an ever increasing series of data $y_1 < \cdots < y_d$, for example $\mathbf{y} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$ the constrained optimal segmentation $\tilde{\mathbf{m}}^3(\mathbf{y})$ is undefined, and the cDPA thus returns no model.

4.3. The cDPA is a heuristic

In this section we give a simple counter-example which shows that the cDPA is not guaranteed to recover a valid **PeakSeg** model even if there is one. This is mostly a theoretical issue, since the cDPA is empirically able to compute **PeakSeg** models in the vast majority of real data sets (Section 6.1).

The cDPA is a heuristic since it is not guaranteed to compute the **PeakSeg** model. We illustrate this using the following example with $\mathbf{y} = \begin{bmatrix} 1 & 10 & 14 & 13 \end{bmatrix} \in \mathbb{Z}_+^4$. For $s = 3$ segments there are only 3 possible segmentations: $[1][10][14, 13]$, $[1][10, 14][13]$ and $[1, 10][14][13]$. If we use max-likelihood estimates for each segment mean, then only the last segmentation obeys the **PeakSeg** model constraints. However the cDPA will not recover it.

That is because the best segmentation recovered by the cDPA for $s = 2$ segments up to point 3 is $[1][10, 14]$. For $s = 3$ the cDPA will then consider the segmentations $[1][10][14, 13]$ and $[1][10, 14][13]$ and discard them since they do not satisfy the **PeakSeg** model constraints. The cDPA will thus report that it didn't find a valid peak model in $s = 3$. However there is one: $[1, 10][14][13]$.

Note that if we run the constrained DP in the backward direction, $\mathbf{y} = \begin{bmatrix} 13 & 14 & 10 & 1 \end{bmatrix} \in \mathbb{Z}_+^4$, we would recover the best solution $[13][14][10, 1]$.

5. Penalty function learning methods

For a profile $\mathbf{y} \in \mathbb{Z}_+^d$ and a penalty constant $\lambda \in \mathbb{R}_+$, the optimal number of segments is

$$s^*(\lambda, \mathbf{y}) = \arg\min_{s \in \{1,3,\dots,s_{\max}\}} \rho\left[\tilde{\mathbf{m}}^s(\mathbf{y}), \mathbf{y}\right] + h(s, d)\lambda, \tag{12}$$

where $h(s, d)$ is a model complexity function that is given. In this paper we consider two types of model complexity h functions: AIC/BIC and oracle (Table 1).

For each training sample $i \in \{1, \dots, n\}$, we have a feature vector $\mathbf{x}_i \in \mathbb{R}^m$ (details in Section 5.3). We learn a penalty function $f(\mathbf{x}_i) = \log \lambda_i$ that predicts a sample-specific number of segments $s^*(\lambda_i, \mathbf{y}_i)$ with minimum train error.

After learning a penalty function $\hat{f}$ on the training data, we

[1] Implemented as C code in the R package https://github.com/tdhock/PeakSegDP

use the following method to make a prediction on a test sample with profile $\mathbf{y}$ and features $\mathbf{x}$. We compute the predicted penalty $\hat{\lambda} = \exp \hat{f}(\mathbf{x})$, the predicted number of segments $\hat{s} = s^*(\hat{\lambda}, \mathbf{y})$, and the predicted peaks $\mathbf{P}\left[\tilde{\mathbf{m}}^{\hat{s}}(\mathbf{y})\right]$.

5.1. Unsupervised penalty functions

There are several unsupervised penalty functions that are based on theoretical assumptions about the data (distribution, independence) which may or may not be true in practice. One can use asymptotic arguments to get a penalty such as the AIC (Akaike, 1973), BIC (Schwarz, 1978), or mBIC (Zhang and Siegmund, 2007). For example, the AIC corresponds to a constant penalty of $\lambda = \exp f(\mathbf{x}) = 2$ and a model complexity of $h(s, d) = s$. More recently, Cleynen and Lebarbier (2014) applied finite sample model selection theory to obtain a more complicated model complexity function h (Table 1) and a heuristic for computing the penalty λ. All of these penalty functions are completely unsupervised since they ignore the annotated region labels in the training data set.

In Table 1 and Figure 3, these models are named *.0 since they do not learn any parameters using the training data.

5.2. Supervised single-parameter penalty learning

Cleynen and Lebarbier (2014) proposed to use the unsupervised heuristic of Arlot and Massart (2009) for calibrating the penalty constant β in equation (6) of their paper. Instead, we propose to use the annotated region labels as a supervised method for calibrating the penalty constant β. This corresponds to learning a constant penalty function $\exp f(\mathbf{x}_i) = \beta = \lambda_i$ for all samples i.

More specifically, we defined a grid of 200 β penalty constants evenly spaced on the log scale between 10^{-2} and 10^4, then used grid search to select the value that minimizes the annotation error (1) on the train set.

In Table 1 and Figure 3, these models are named *.1 since they each learn 1 parameter using the training data.

5.3. Supervised multi-parameter penalty learning

For supervised learning of multi-parameter penalties, we define sample-specific penalty values $\log \lambda_i = f(\mathbf{x}_i) = \beta + \mathbf{w}^\intercal \mathbf{x}_i$, which is an affine function with parameters $\beta \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^m$ that will be learned. As shown in Table 1, we considered learning models with 3 and 41 parameters:

- **3**: we used an $m = 2$-dimensional feature vector $\mathbf{x}_i = \left[\begin{array}{cc} \log\max \mathbf{y}_i & \log d_i \end{array}\right]$ where $\mathbf{y}_i \in \mathbb{Z}_+^{d_i}$ is one sample i of count data to segment. This corresponds to a penalty $\lambda_i = e^\beta (\max \mathbf{y}_i)^{w_1} d_i^{w_2}$. We fit the model by solving the un-regularized interval regression problem of Rigaill et al. (2013).
- **41**: we defined $m = 40$ features using transforms $(x, \log x, \log[x+1], \log\log x)$, where x are features such as quartiles of $\mathbf{y}_i$, mean of $\mathbf{y}_i$, and number of data points d_i. We fit the model by solving the L1-regularized problem of Rigaill et al. (2013).

6. Results: state-of-the-art peak detection for two patterns

We downloaded 7 benchmark data sets, which included a total of 12,826 manually annotated regions across 2,752 separate segmentation problems.[2] Data sets span two different data/pattern types (sharp H3K4me3, broad H3K36me3), four annotators (AM, TDH, PGP, XJ), and two cell type groups (immune, other). Annotators are the PhD students and post-docs who created the labels via visual inspection of genome browser plots such as Figure 1. Each data set contains labels grouped into windows of nearby regions (from 4 to 30 windows per data set). For each data set, we performed 6 random splits of windows into half train, half test. We considered 3 kinds of model training:[3]

- **Unsupervised** uses the default parameters of each algorithm, ignoring the labeled data in the train set.
- **Grid search** learns a scalar parameter with minimal annotation error on the train set.
- **Interval regression** is the multi-parameter penalty learning algorithm of Rigaill et al. (2013).

6.1. Effectiveness of heuristic cDPA

To run the cDPA on the benchmark data set, we first set the maximum number of segments $s_{\max} = 19$, meaning a maximum of 9 peaks. Running the cDPA on all of the data sets in the benchmark took a total of about 6.5 days. For the largest segmentation problem we considered ($d = 263,169$ data points), the cDPA took about 155 minutes.

As discussed in Section 4.3, the cDPA is a heuristic for computing the **PeakSeg** model. To assess whether or not this is a problem in real data sets, we checked for how many segmentation problems the cDPA returned the complete sequence of 10 models $\tilde{\mathbf{m}}^1(\mathbf{y}), \tilde{\mathbf{m}}^3(\mathbf{y}), \ldots, \tilde{\mathbf{m}}^{19}(\mathbf{y})$. For the vast majority of segmentation problems ($2738/2752 = 99.5\%$), the constrained DP returned all 10 models. For the other 14 problems, the algorithm did not return at least one of the ten models. Of these 14 problems with missing

[2] http://cbio.ensmp.fr/~thocking/chip-seq-chunk-db/

[3] Source code for computing models and benchmarks at https://github.com/tdhock/PeakSeg-paper

models, 11 problems still had at least one perfect peak detection model with zero annotation error (1). We concluded that although the cDPA is technically a heuristic algorithm for solving **PeakSeg**, it is still a useful peak detector in the vast majority of real data sets.

6.2. Comparison of unsupervised methods

In a previous study of this benchmark data set (Hocking et al., 2014), the macs algorithm was found to be the most accurate peak detector for H3K4me3 data, whereas hmcan.broad was best for H3K36me3 data.

Figure 3 shows that in three H3K36me3 data sets (broad peak pattern), hmcan.broad.0 is clearly more accurate than macs.0. Furthermore, it is clear that the oracle.0 model is at least as accurate as hmcan.broad.0.

For the four H3K4me3 data sets (sharp peak pattern), oracle.0 is about as accurate as macs.0, which is clearly more accurate than hmcan.broad.0.

Finally, the unsupervised AIC/BIC.0 method is clearly the least accurate of all methods, since it always picks the model with the most peaks (same as no penalty, for both AIC and BIC). We also tried the modified BIC of Zhang and Siegmund (2007), but obtained the same high false positive rates.

6.3. Supervised learning of 1 parameter increases peak detection accuracy

Does training a single scalar parameter using grid search increase peak detection accuracy relative to unsupervised default models? Yes, in the vast majority of cases. Figure 3 shows that supervised *.1 models have generally lower test error than the corresponding unsupervised *.0 models.

name	**model complexity** $h(s, d_i)$
AIC/BIC.*	s
oracle.*	$s\left(1 + 4\sqrt{1.1 + \log(d_i/s)}\right)^2$

name	**smoothing** λ_i	**parameters**	**learning algorithm**
*.0	AIC=2, BIC=$\log d_i$	none	unsupervised
*.1	β	$\beta \in \mathbb{R}_+$	grid search
*.3	$e^{\beta} d_i^{w_1} (\max \mathbf{y}_i)^{w_2}$	$\beta, w_1, w_2 \in \mathbb{R}$	interval regression
*.41	$\exp(\beta + \mathbf{w}^\intercal \mathbf{x}_i)$	$\beta \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^{40}$	regularized int. reg.

Table 1. Penalties for model selection (12) are of the form $h(s, d_i)\lambda_i$ for data to segment $\mathbf{y}_i \in \mathbb{Z}_+^{d_i}$ and models with s segments. Names show model complexity $h(s, d_i)$ (left) and number of parameters to learn in λ_i (right). For example, the AIC/BIC.41 model has a penalty of $s \exp(\beta + \mathbf{w}^\intercal \mathbf{x}_i)$, where the parameters $\beta, \mathbf{w}$ are learned using regularized interval regression on a training data set.

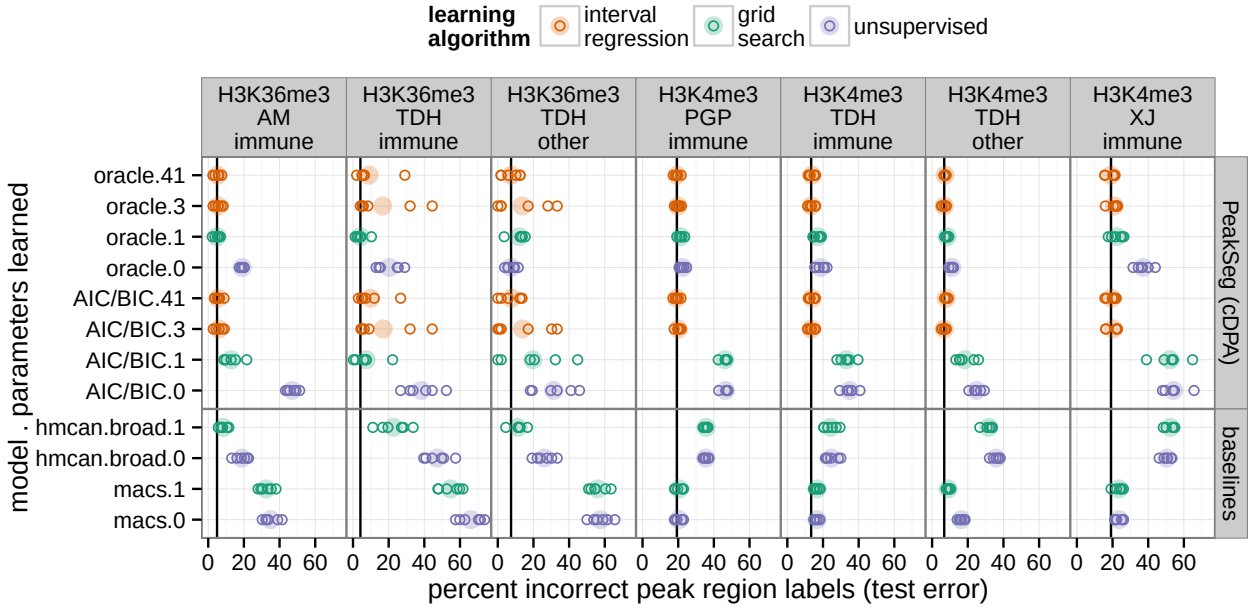

Figure 3. Test error of peak detection algorithms on seven labeled data sets (each point shows one of six randomly selected train/test splits, the shaded circle is the mean test error, and the vertical black line is the mean of the best algorithm for each data set). Data set names show data/pattern type (e.g. H3K36me3), annotator (AM), and cell types (immune). Colors show how the training data were used to learn model parameters.

Among models with only 1 parameter trained using grid search, it is clear that oracle.1 is the most accurate. It achieves close to state-of-the-art peak detection accuracy on every data set. In contrast, the baseline macs.1 is only effective for H3K4me3 data, and the baseline hmcan.broad.1 is only effective for H3K36me3 data.

6.4. Supervised learning of several parameters can further increase peak detection accuracy

Does supervised multi-parameter model training improve over single-parameter models? A little, if there are sufficient training data. Figure 3 shows that multi-parameter models (*.3 and *.41) tend to be at least as accurate as the corresponding single-parameter *.1 model. The only exceptions are the H3K36me3_TDH data sets, for which there are few training data (only 2 groups of labeled regions per train set). For these data, the *.3 and *.41 models sometimes overfit relative to the corresponding *.1 model.

7. Discussion

7.1. Supervised versus unsupervised learning

Baseline unsupervised peak detectors in the bioinformatics literature are each designed to recognize a specific peak pattern. For example we observed that the macs algorithm was effective for the sharp peak pattern in H3K4me3 data, and the hmcan.broad algorithm was effective for the broad peak pattern in H3K36me3.

In contrast, we propose a supervised learning approach for peak detection. Rather than designing a specific model for each pattern, we propose to use a database of labels with a general peak detector that can be trained to recognize different patterns. We showed that supervised penalty learning with the **PeakSeg** model can be used for accurate peak detection in both H3K4me3 and H3K36me3 data.

In general, it is clear that the supervised learning methods were more accurate than their unsupervised counterparts. The only exception is that sometimes when there are few training data, the multi-parameter supervised learning methods overfit (Figure 3, H3K36me_TDH data sets). So when there are very few training data, we recommend using the oracle.1 supervised model which avoids overfitting by learning just 1 parameter using grid search. However, when there are many training data, we would recommend using a multi-parameter supervised learning method such as oracle.41.

7.2. Oracle penalty is more accurate than AIC/BIC

We tested the peak detection accuracy of two types of model complexity h functions (Table 1). Among unsupervised models, Figure 3 shows that the oracle.0 model is always more accurate than the AIC/BIC.0 model. For the supervised 1-parameter models, it is also clear that oracle.1 is more accurate than AIC/BIC.1. However, we obtained similar test error rates when we used multi-parameter supervised learning with AIC/BIC and oracle penalties.

In general, these data provide convincing evidence that the model selection theory of Cleynen and Lebarbier (2014) is useful for accurate peak detection in real ChIP-seq data sets, especially when combined with supervised penalty learning methods.

8. Conclusions and future work

We proposed the **PeakSeg** constrained maximum likelihood segmentation model as a peak detector in ChIP-seq count data. We proposed a constrained dynamic programming algorithm that efficiently computes a sequence of segmentations that satisfy the **PeakSeg** constraint. Furthermore, we proposed to use annotated region labels as supervision in a penalty learning problem. Whereas unsupervised baseline methods are only effective for a single data/pattern type, our approach yields state-of-the-art peak detection in a benchmark that includes both sharp H3K4me3 and broad H3K36me3 data/pattern types.

There are several modeling choices that could be explored to increase peak detection accuracy. First, we could replace the Poisson loss of the **PeakSeg** model with a negative binomial loss (Cleynen and Lebarbier, 2014), which has an additional variance parameter. Also, a more accurate model could possibly be obtained by engineering better features $\mathbf{x}_i$, learning a non-linear smoothing function f, or learning model complexity functions h. For example the oracle penalty of Cleynen and Lebarbier (2014) has an assumed constant of 1.1 (Table 1), which could be learned instead. Finally, we could explore convex relaxations similar to the Fused Lasso (Hoefling, 2009), by replacing the sign function with the positive part function in (6).

The current implementation of **PeakSeg** using the cDPA has several limitations. First, the $O(d^2)$ time complexity for d data points could be easily reduced to $O(d \log d)$ by using a constrained version of pruned dynamic programming (Rigaill, 2010; Cleynen et al., 2014). Second, we showed that the cDPA is a heuristic for computing the **PeakSeg** model (Section 4.3). On one hand, we would like to know what optimization problem the cDPA solves. On the other hand, we would like an efficient algorithm that exactly computes the **PeakSeg** model.

Finally, we are interested in jointly segmenting multiple samples at the same time, since peaks are often observed in the same genomic location across several samples of the same cell type.

References

H. Akaike. Information theory as an extension of the maximum likelihood principle. In B. Petrov and F. Csaki, editors, *Second International Symposium on Information Theory*, pages 267–281. Akademiai Kiado, Budapest, 1973.

S. Arlot and P. Massart. Data-driven calibration of penalties for least-squares regression. *The Journal of Machine Learning Research*, 10:245–279, 2009.

H. Ashoor, A. Hérault, A. Kamoun, F. Radvanyi, V. B. Bajic, E. Barillot, and V. Boeva. HMCan: a method for detecting chromatin modifications in cancer samples using ChIP-seq data. *Bioinformatics*, 29(23):2979–2986, 2013.

T. Bailey, P. Krajewski, I. Ladunga, C. Lefebvre, Q. Li, T. Liu, P. Madrigal, C. Taslim, and J. Zhang. Practical guidelines for the comprehensive analysis of ChIP-seq data. *PLoS computational biology*, 9(11):e1003326, 2013.

R. Bellman. On the approximation of curves by line segments using dynamic programming. *Commun. ACM*, 4 (6):284–, June 1961.

A. Cleynen, M. Koskas, E. Lebarbier, G. Rigaill, and S. Robin. Segmentor3IsBack: an R package for the fast and exact segmentation of Seq-data. *Algorithms for Molecular Biology*, 9:6, 2014.

T. D. Hocking, P. Goerner-Potvin, A. Morin, X. Shao, and G. Bourque. Visual annotations and a supervised learning approach for evaluating and calibrating ChIP-seq peak detectors. *arXiv:1409.6209*, 2014.

H. Hoefling. A path algorithm for the Fused Lasso Signal Approximator. arXiv:0910.0526, 2009.

V. Kumar, M. Muratani, N. A. Rayan, P. Kraus, T. Lufkin, H. H. Ng, and S. Prabhakar. Uniform, optimal signal processing of mapped deep-sequencing data. *Nature biotechnology*, 31(7):615–622, 2013.

A. Cleynen and E. Lebarbier. Segmentation of the Poisson and negative binomial rate models: a penalized estimator. *ESAIM: PS*, 18:750–769, 2014.

M. Micsinai, F. Parisi, F. Strino, P. Asp, B. D. Dynlacht, and Y. Kluger. Picking ChIP-seq peak detectors for analyzing chromatin modification experiments. *Nucleic acids research*, page gks048, 2012.

G. Rigaill. Pruned dynamic programming for optimal multiple change-point detection. arXiv:1004.0887, 2010.

G. Rigaill, T. Hocking, J.-P. Vert, and F. Bach. Learning sparse penalties for change-point detection using max margin interval regression. In *Proc. 30th ICML*, pages 172–180, 2013.

M. B. Rye, P. Sætrom, and F. Drabløs. A manually curated ChIP-seq benchmark demonstrates room for improvement in current peak-finder programs. *Nucleic acids research*, page gkq1187, 2010.

G. Schwarz. Estimating the dimension of a model. *Ann. Statist.*, 6(2):461–464, 1978.

A. M. Szalkowski and C. D. Schmid. Rapid innovation in chip-seq peak-calling algorithms is outdistancing benchmarking efforts. *Briefings in Bioinformatics*, 12(6):626–633, 2011.

E. Wilibanks and M. Facciotti. Evaluation of Algorithm Performance in ChIP-Seq Peak Detection. *PLoS ONE*, 5 (7), 2010.

Y.-C. Yao. Estimating the number of change-points via Schwarz' criterion. *Statistics & Probability Letters*, 6 (3):181–189, February 1988.

N. R. Zhang and D. O. Siegmund. A Modified Bayes Information Criterion with Applications to the Analysis of Comparative Genomic Hybridization Data. *Biometrics*, 63:22–32, 2007.

Y. Zhang, T. Liu, C. A. Meyer, J. Eeckhoute, D. S. Johnson, B. E. Bernstein, C. Nusbaum, R. M. Myers, M. Brown, W. Li, et al. Model-based analysis of ChIP-Seq (MACS). *Genome Biol*, 9(9):R137, 2008.