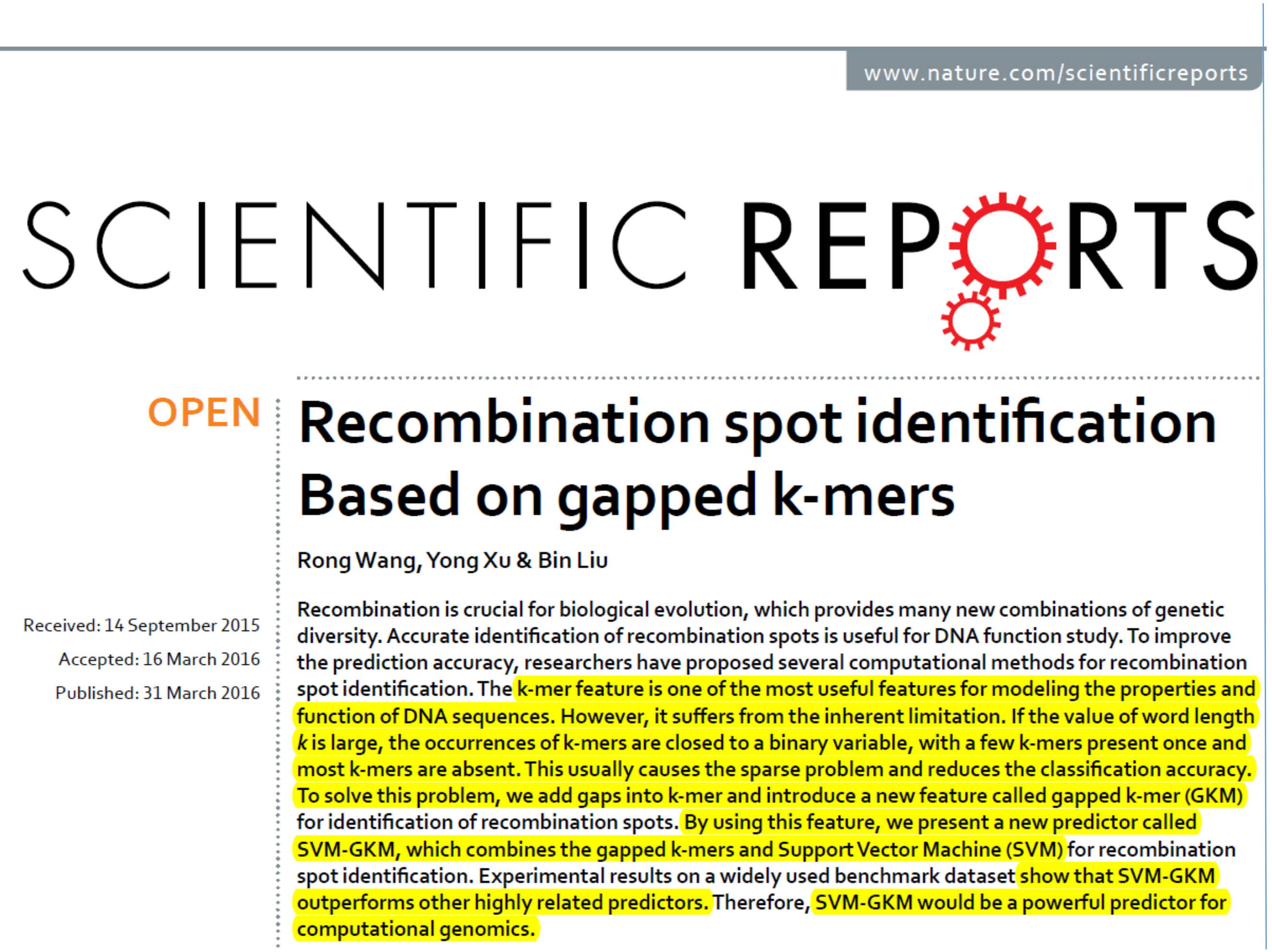


www.nature.com/scientificreports

SCIENTIFIC REPORTS

OPEN

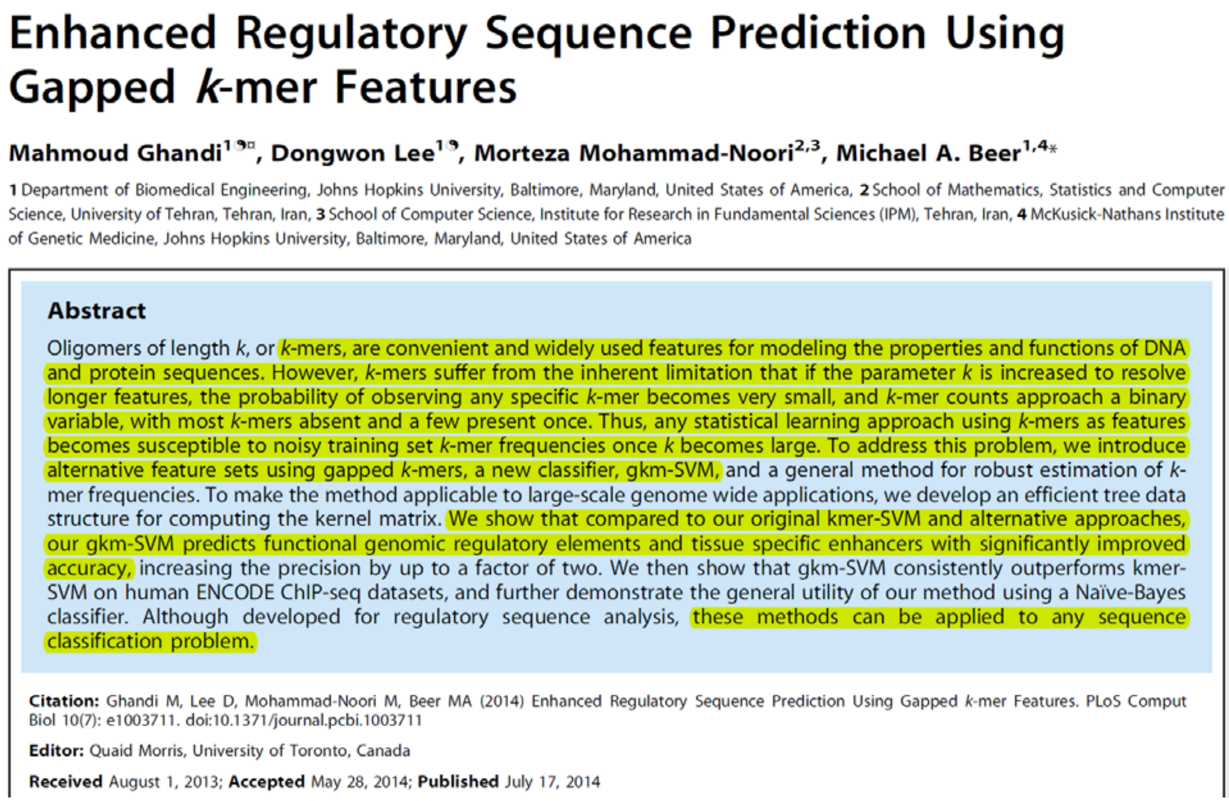
Recombination spot identification Based on gapped k-mers

Rong Wang, Yong Xu & Bin Liu

Received: 14 September 2015
Accepted: 16 March 2016
Published: 31 March 2016

Recombination is crucial for biological evolution, which provides many new combinations of genetic diversity. Accurate identification of recombination spots is useful for DNA function study. To improve the prediction accuracy, researchers have proposed several computational methods for recombination spot identification. The k-mer feature is one of the most useful features for modeling the properties and function of DNA sequences. However, it suffers from the inherent limitation. If the value of word length *k* is large, the occurrences of k-mers are closed to a binary variable, with a few k-mers present once and most k-mers are absent. This usually causes the sparse problem and reduces the classification accuracy. To solve this problem, we add gaps into k-mer and introduce a new feature called gapped k-mer (GKM) for identification of recombination spots. By using this feature, we present a new predictor called SVM-GKM, which combines the gapped k-mers and Support Vector Machine (SVM) for recombination spot identification. Experimental results on a widely used benchmark dataset show that SVM-GKM outperforms other highly related predictors. Therefore, SVM-GKM would be a powerful predictor for computational genomics.

original article:



Enhanced Regulatory Sequence Prediction Using Gapped *k*-mer Features

Mahmoud Ghandi[1☯¤], Dongwon Lee[1☯], Morteza Mohammad-Noori[2,3], Michael A. Beer[1,4*]

1 Department of Biomedical Engineering, Johns Hopkins University, Baltimore, Maryland, United States of America, 2 School of Mathematics, Statistics and Computer Science, University of Tehran, Tehran, Iran, 3 School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran, 4 McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins University, Baltimore, Maryland, United States of America

Abstract

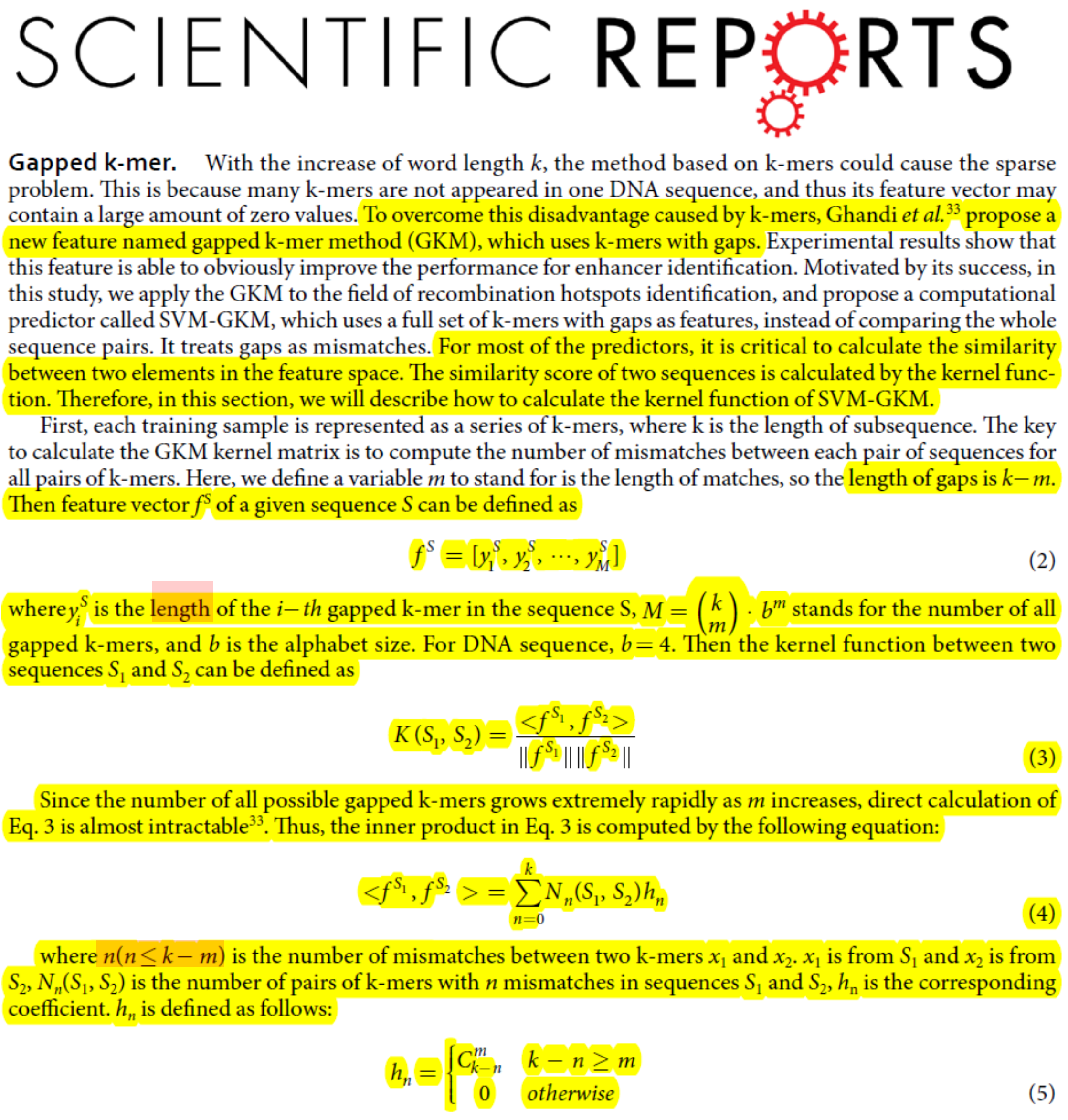
Oligomers of length *k*, or *k*-mers, are convenient and widely used features for modeling the properties and functions of DNA and protein sequences. However, *k*-mers suffer from the inherent limitation that if the parameter *k* is increased to resolve longer features, the probability of observing any specific *k*-mer becomes very small, and *k*-mer counts approach a binary variable, with most *k*-mers absent and a few present once. Thus, any statistical learning approach using *k*-mers as features becomes susceptible to noisy training set *k*-mer frequencies once *k* becomes large. To address this problem, we introduce alternative feature sets using gapped *k*-mers, a new classifier, gkm-SVM, and a general method for robust estimation of *k*-mer frequencies. To make the method applicable to large-scale genome wide applications, we develop an efficient tree data structure for computing the kernel matrix. We show that compared to our original kmer-SVM and alternative approaches, our gkm-SVM predicts functional genomic regulatory elements and tissue specific enhancers with significantly improved accuracy, increasing the precision by up to a factor of two. We then show that gkm-SVM consistently outperforms kmer-SVM on human ENCODE ChIP-seq datasets, and further demonstrate the general utility of our method using a Naïve-Bayes classifier. Although developed for regulatory sequence analysis, these methods can be applied to any sequence classification problem.

Citation: Ghandi M, Lee D, Mohammad-Noori M, Beer MA (2014) Enhanced Regulatory Sequence Prediction Using Gapped *k*-mer Features. PLoS Comput Biol 10(7): e1003711. doi:10.1371/journal.pcbi.1003711

Editor: Quaid Morris, University of Toronto, Canada

Received August 1, 2013; **Accepted** May 28, 2014; **Published** July 17, 2014

Highly similar sections are highlighted. While the 2014 PLOS Comp Bio paper is cited, the 2016 Scientific Reports article incorrectly claims "we... introduce a new feature called gapped k-mer" and "we present a new predictor called (SVM-GKM)," which were introduced and developed in the 2014 PLOS Comp Bio paper as gkm-SVM. The "SVM-GKM" method is identical to gkm-SVM, and uses our gkm-SVM software. The authors have copied text, rearranged the acronym, rearranged some variable names (introducing some errors in the process), and run our software on a different dataset. The claim of this paper was not "we applied gkm-SVM to recombination", the fraudulent claim was "we present a new predictor called (SVM-GKM)."



SCIENTIFIC REPORTS

Gapped k-mer. With the increase of word length k, the method based on k-mers could cause the sparse problem. This is because many k-mers are not appeared in one DNA sequence, and thus its feature vector may contain a large amount of zero values. To overcome this disadvantage caused by k-mers, Ghandi *et al.*[33] propose a new feature named gapped k-mer method (GKM), which uses k-mers with gaps. Experimental results show that this feature is able to obviously improve the performance for enhancer identification. Motivated by its success, in this study, we apply the GKM to the field of recombination hotspots identification, and propose a computational predictor called SVM-GKM, which uses a full set of k-mers with gaps as features, instead of comparing the whole sequence pairs. It treats gaps as mismatches. For most of the predictors, it is critical to calculate the similarity between two elements in the feature space. The similarity score of two sequences is calculated by the kernel function. Therefore, in this section, we will describe how to calculate the kernel function of SVM-GKM.

First, each training sample is represented as a series of k-mers, where k is the length of subsequence. The key to calculate the GKM kernel matrix is to compute the number of mismatches between each pair of sequences for all pairs of k-mers. Here, we define a variable m to stand for is the length of matches, so the length of gaps is $k-m$. Then feature vector f^S of a given sequence S can be defined as

$$f^S = [y_1^S, y_2^S, \cdots, y_M^S] \tag{2}$$

where y_i^S is the length of the $i-th$ gapped k-mer in the sequence S, $M = \binom{k}{m} \cdot b^m$ stands for the number of all gapped k-mers, and b is the alphabet size. For DNA sequence, $b=4$. Then the kernel function between two sequences S_1 and S_2 can be defined as

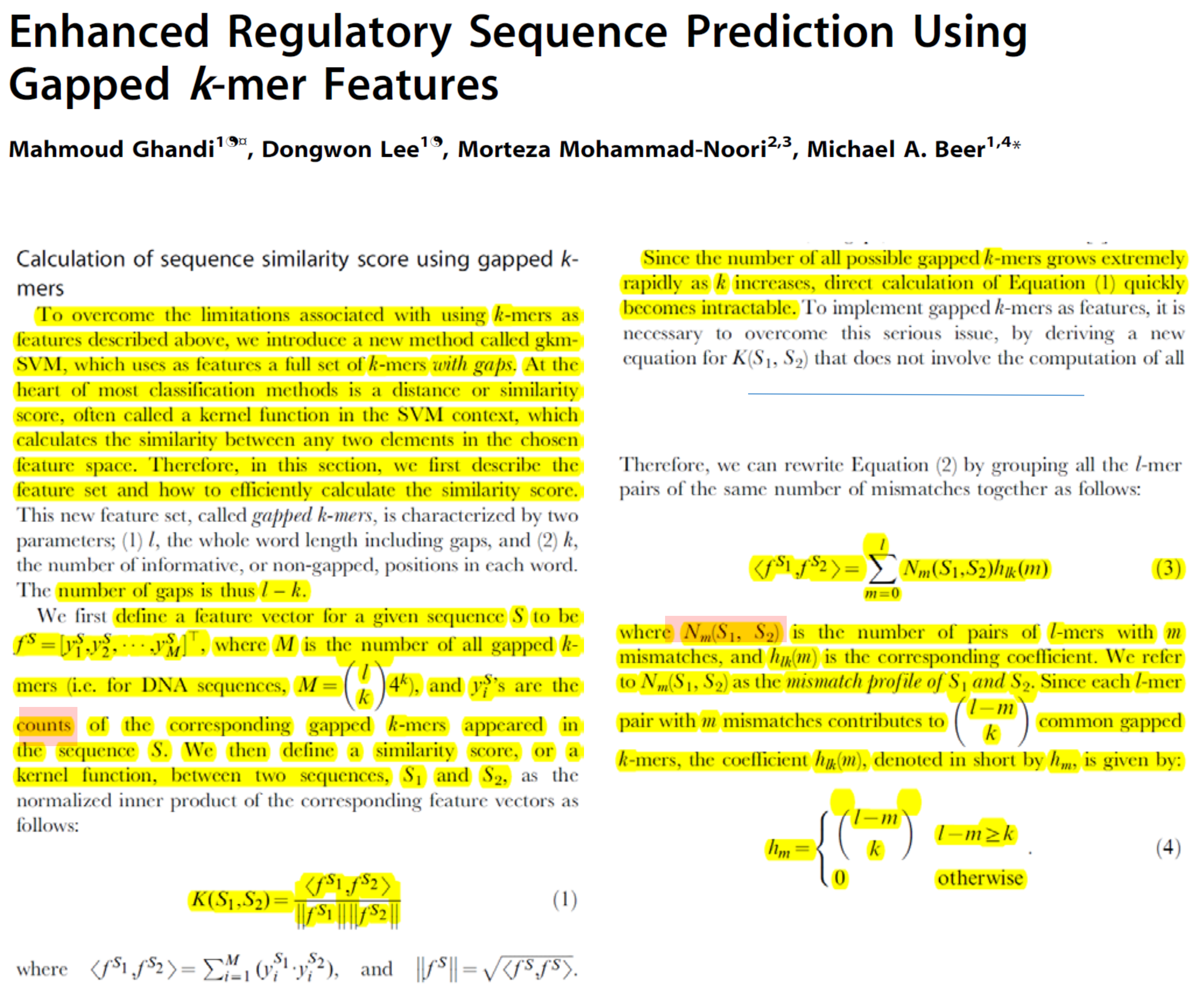
$$K(S_1, S_2) = \frac{<f^{S_1}, f^{S_2}>}{\|f^{S_1}\| \|f^{S_2}\|} \tag{3}$$

Since the number of all possible gapped k-mers grows extremely rapidly as m increases, direct calculation of Eq. 3 is almost intractable[33]. Thus, the inner product in Eq. 3 is computed by the following equation:

$$<f^{S_1}, f^{S_2}> = \sum_{n=0}^{k} N_n(S_1, S_2) h_n \tag{4}$$

where $n(n \le k-m)$ is the number of mismatches between two k-mers x_1 and x_2. x_1 is from S_1 and x_2 is from S_2, $N_n(S_1, S_2)$ is the number of pairs of k-mers with n mismatches in sequences S_1 and S_2, h_n is the corresponding coefficient. h_n is defined as follows:

$$h_n = \begin{cases} C_{k-n}^{m} & k-n \ge m \\ 0 & otherwise \end{cases} \tag{5}$$



Enhanced Regulatory Sequence Prediction Using Gapped *k*-mer Features

Mahmoud Ghandi[1¶¤], Dongwon Lee[1¶], Morteza Mohammad-Noori[2,3], Michael A. Beer[1,4*]

Calculation of sequence similarity score using gapped *k*-mers

To overcome the limitations associated with using *k*-mers as features described above, we introduce a new method called gkm-SVM, which uses as features a full set of *k*-mers *with gaps*. At the heart of most classification methods is a distance or similarity score, often called a kernel function in the SVM context, which calculates the similarity between any two elements in the chosen feature space. Therefore, in this section, we first describe the feature set and how to efficiently calculate the similarity score. This new feature set, called *gapped k-mers*, is characterized by two parameters; (1) l, the whole word length including gaps, and (2) k, the number of informative, or non-gapped, positions in each word. The number of gaps is thus $l-k$.

We first define a feature vector for a given sequence S to be $f^S = [y_1^S, y_2^S, \cdots, y_M^S]^\top$, where M is the number of all gapped k-mers (i.e. for DNA sequences, $M = \binom{l}{k} 4^k$), and y_i^S's are the counts of the corresponding gapped k-mers appeared in the sequence S. We then define a similarity score, or a kernel function, between two sequences, S_1 and S_2, as the normalized inner product of the corresponding feature vectors as follows:

$$K(S_1, S_2) = \frac{\langle f^{S_1}, f^{S_2} \rangle}{\|f^{S_1}\| \|f^{S_2}\|} \tag{1}$$

where $\langle f^{S_1}, f^{S_2} \rangle = \sum_{i=1}^{M} (y_i^{S_1} \cdot y_i^{S_2})$, and $\|f^S\| = \sqrt{\langle f^S, f^S \rangle}$.

Since the number of all possible gapped k-mers grows extremely rapidly as k increases, direct calculation of Equation (1) quickly becomes intractable. To implement gapped k-mers as features, it is necessary to overcome this serious issue, by deriving a new equation for $K(S_1, S_2)$ that does not involve the computation of all

Therefore, we can rewrite Equation (2) by grouping all the l-mer pairs of the same number of mismatches together as follows:

$$\langle f^{S_1}, f^{S_2} \rangle = \sum_{m=0}^{l} N_m(S_1, S_2) h_{lk}(m) \tag{3}$$

where $N_m(S_1, S_2)$ is the number of pairs of l-mers with m mismatches, and $h_{lk}(m)$ is the corresponding coefficient. We refer to $N_m(S_1, S_2)$ as the *mismatch profile of* S_1 *and* S_2. Since each l-mer pair with m mismatches contributes to $\binom{l-m}{k}$ common gapped k-mers, the coefficient $h_{lk}(m)$, denoted in short by h_m, is given by:

$$h_m = \begin{cases} \binom{l-m}{k} & l-m \ge k \\ 0 & \text{otherwise} \end{cases}. \tag{4}$$

These equations are all identical, with only variable name and notation changes: b=4, k→l, n→m, m→k, $C_{k-n}^{m} = \binom{k-n}{m}$.
Copying errors in red: "counts" was changed to "length," which would make this method fail, if it had actually been implemented.

SCIENTIFIC REPORTS

In order to reduce the error caused by corresponding coefficients, the following equation is used to get h_n when calculating the mismatch for two sequences

$$h_n = \sum_{n_1=0}^{M} \sum_{n_2=0}^{M} \sum_{t=0}^{M} C_{k-n}^{t}(b-1)^t C_n^r (b-2)^r C_{n-r}^{n_1-t-r} \qquad (6)$$

where n_1 is the mismatch number that k-mer x_1 contains, n_2 is the mismatch number that k-mer x_2 contains, and t is the mismatches number, which exists at the $k-n$ mismatch positions for both x_1 and x_2. The remaining mismatches $r = n_2 - t - (n - n_1 - t)$ are among the the n mismatch positions for k-mer x_2.

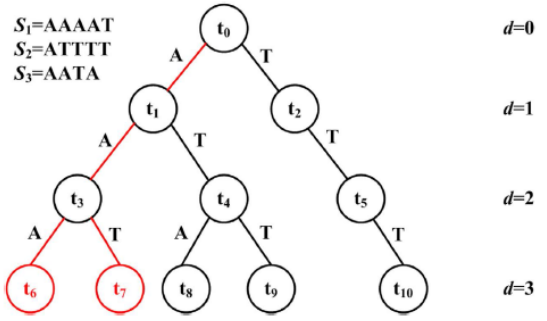


Figure 1. An example to show the tree structure of k-mer counting. This example only contains two alphabets, A and T. We use $k = 3$ and three sequences S_1 = AAAAT, S_2 = ATTTT, and S_3 = AATA to build k-mer tree. Each node t_i at depth d represents a sequence of length d, denoted by $s(t_i)$, which is determined by the path from the root of the tree to t_i. At depth $d = 3$, for node t_6, $s(t_6)$ = 'AAA', S_1 contains two counts of this k-mer, S_2 and S_3 do not contain this k-mer. For node t_7, $s(t_7)$ = 'AAT', S_1 and S_3 both contain one count, and S_2 does not contain this k-mer. Compared t_6 with t_7, the paths to these two nodes only contain one mismatch.

Enhanced Regulatory Sequence Prediction Using Gapped *k*-mer Features

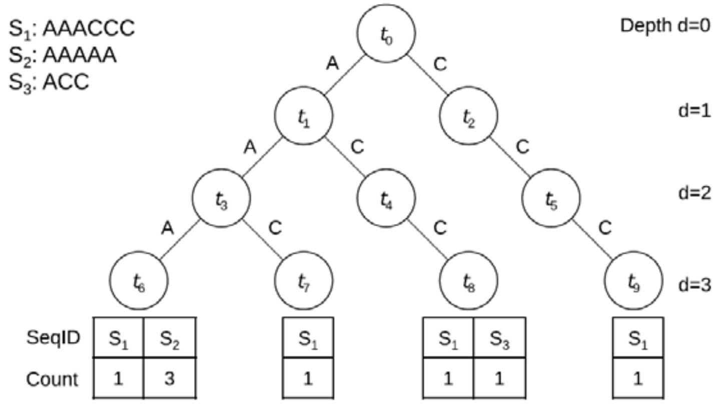
Mahmoud Ghandi[1¶¤], Dongwon Lee[1¶], Morteza Mohammad-Noori[2,3], Michael A. Beer[1,4*]

used in Equation (3). To calculate the mismatch string kernel value for two sequences we replace $h_{lk}(m)$ in Equation (3) by $h_{l,M}^{mismatch}(m)$:

$$h_{lM}^{mismatch}(m) = \sum_{m_1=0}^{M} \sum_{m_2=0}^{M} \sum_{t=0}^{M} \binom{l-m}{t}(b-1)^t \binom{m}{r}(b-2)^r \binom{m-r}{m_1-t-r} \qquad (9)$$

where b is the alphabet size ($b=4$ for DNA sequences) and $r = m_1 + m_2 - m - 2t$. Given two l-mers x_1 and x_2 where x_1 and x_2 differ in exactly m places, the term inside the summations counts the number of all possible l-mers that exactly differ x_1 in m_1 places and x_2 in m_2 places t of which fall in the common l-m bases of x_1 and x_2. (See Figure S9). So the result of the summation is the number

here when retyping they introduced an error in the definition of r.

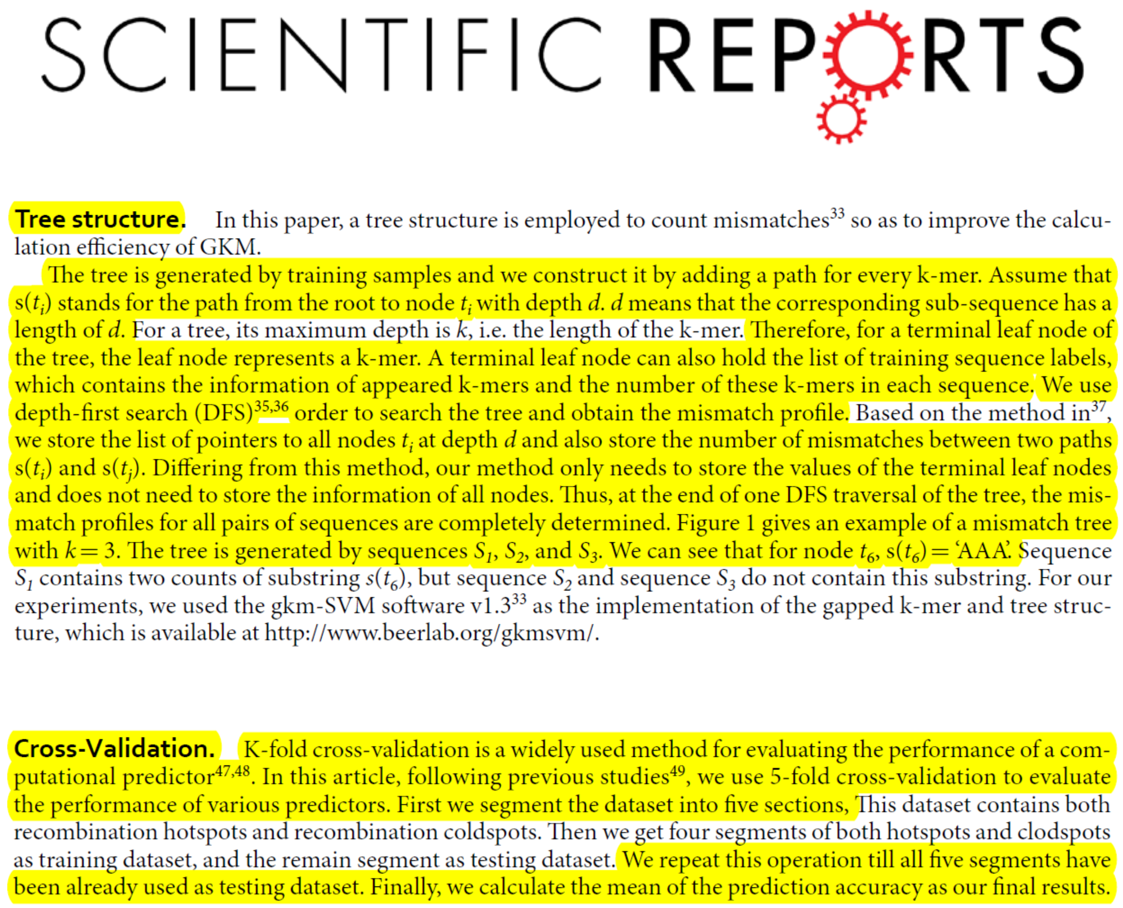


SeqID	S_1	S_2	S_1	S_1	S_3	S_1
Count	1	3	1	1	1	1

```
DFS(node, depth, {(node, mismatches)})
DFS(t_0 , 0, {(t_0, 0)})
    DFS(t_1 , 1, {(t_1, 0), (t_2, 1)})
        DFS(t_3 , 2, {(t_3, 0), (t_4, 1), (t_5, 2)})
            DFS(t_6 , 3, {(t_6, 0), (t_7, 1), (t_8, 2), (t_9, 3)})
            DFS(t_7 , 3, {(t_6, 1), (t_7, 0), (t_8, 1), (t_9, 2)})
        ...
    DFS(t_2 , 1, {(t_1, 1), (t_2, 0)})
        ...
```

changed C→ T algorithm is identical

Figure 5. Fast computation of mismatch profiles using *k*-mer tree structure. As an example, we use $l=3$ and three sequences S_1 = AAACCC, S_2 = AAAAA, and S_3 = ACC to build the *k*-mer tree. The leaves (nodes at depth $d = l = 3$) correspond to 3-mers AAA, AAC, ACC, and CCC. The sequence ID and the number of times each 3-mer appeared in each sequence are stored for each leaf. Each node t_i at depth d represents a sequence of length d, denoted by $s(t_i)$, which is determined by the path from the root of the tree to t_i. For example, $s(t_2)$ = C and $s(t_4)$ = AC. DFS is started at the root node, t_0. When visiting each node t_i, at depth d, we compute the list of all the nodes t_j at depth d for which $s(t_i)$ and $s(t_j)$ have at most m_{max} mismatches. We also compute the number of mismatches between $s(t_i)$ and $s(t_j)$. When reaching a leaf, we increment the corresponding mismatch profile $N_m(S_i, S_j)$ for each pair of sequences S_i in that leaf and S_j in the list.

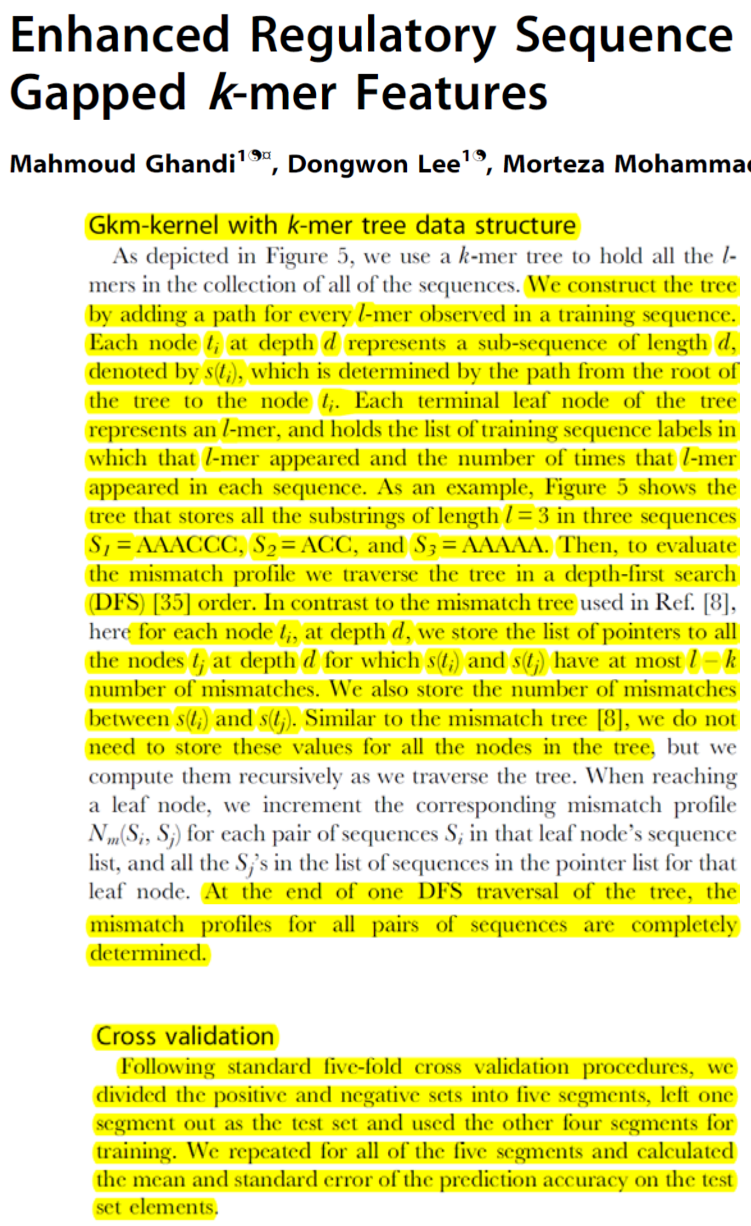


SCIENTIFIC REPORTS

Tree structure. In this paper, a tree structure is employed to count mismatches[33] so as to improve the calculation efficiency of GKM.

The tree is generated by training samples and we construct it by adding a path for every k-mer. Assume that s(t_i) stands for the path from the root to node t_i with depth *d*. *d* means that the corresponding sub-sequence has a length of *d*. For a tree, its maximum depth is *k*, i.e. the length of the k-mer. Therefore, for a terminal leaf node of the tree, the leaf node represents a k-mer. A terminal leaf node can also hold the list of training sequence labels, which contains the information of appeared k-mers and the number of these k-mers in each sequence. We use depth-first search (DFS)[35,36] order to search the tree and obtain the mismatch profile. Based on the method in[37], we store the list of pointers to all nodes t_i at depth *d* and also store the number of mismatches between two paths s(t_i) and s(t_j). Differing from this method, our method only needs to store the values of the terminal leaf nodes and does not need to store the information of all nodes. Thus, at the end of one DFS traversal of the tree, the mismatch profiles for all pairs of sequences are completely determined. Figure 1 gives an example of a mismatch tree with $k = 3$. The tree is generated by sequences S_1, S_2, and S_3. We can see that for node t_6, s(t_6) = 'AAA'. Sequence S_1 contains two counts of substring *s*(t_6), but sequence S_2 and sequence S_3 do not contain this substring. For our experiments, we used the gkm-SVM software v1.3[33] as the implementation of the gapped k-mer and tree structure, which is available at http://www.beerlab.org/gkmsvm/.

Cross-Validation. K-fold cross-validation is a widely used method for evaluating the performance of a computational predictor[47,48]. In this article, following previous studies[49], we use 5-fold cross-validation to evaluate the performance of various predictors. First we segment the dataset into five sections, This dataset contains both recombination hotspots and recombination coldspots. Then we get four segments of both hotspots and clodspots as training dataset, and the remain segment as testing dataset. We repeat this operation till all five segments have been already used as testing dataset. Finally, we calculate the mean of the prediction accuracy as our final results.



Enhanced Regulatory Sequence Prediction Using Gapped *k*-mer Features

Mahmoud Ghandi[1¶¤], **Dongwon Lee**[1¶], **Morteza Mohammad-Noori**[2,3], **Michael A. Beer**[1,4*]

Gkm-kernel with *k*-mer tree data structure

As depicted in Figure 5, we use a *k*-mer tree to hold all the *l*-mers in the collection of all of the sequences. We construct the tree by adding a path for every *l*-mer observed in a training sequence. Each node t_i at depth *d* represents a sub-sequence of length *d*, denoted by $s(t_i)$, which is determined by the path from the root of the tree to the node t_i. Each terminal leaf node of the tree represents an *l*-mer, and holds the list of training sequence labels in which that *l*-mer appeared and the number of times that *l*-mer appeared in each sequence. As an example, Figure 5 shows the tree that stores all the substrings of length $l = 3$ in three sequences S_1 = AAACCC, S_2 = ACC, and S_3 = AAAAA. Then, to evaluate the mismatch profile we traverse the tree in a depth-first search (DFS) [35] order. In contrast to the mismatch tree used in Ref. [8], here for each node t_i, at depth *d*, we store the list of pointers to all the nodes t_j at depth *d* for which $s(t_i)$ and $s(t_j)$ have at most $l - k$ number of mismatches. We also store the number of mismatches between $s(t_i)$ and $s(t_j)$. Similar to the mismatch tree [8], we do not need to store these values for all the nodes in the tree, but we compute them recursively as we traverse the tree. When reaching a leaf node, we increment the corresponding mismatch profile $N_m(S_i, S_j)$ for each pair of sequences S_i in that leaf node's sequence list, and all the S_j's in the list of sequences in the pointer list for that leaf node. At the end of one DFS traversal of the tree, the mismatch profiles for all pairs of sequences are completely determined.

Cross validation

Following standard five-fold cross validation procedures, we divided the positive and negative sets into five segments, left one segment out as the test set and used the other four segments for training. We repeated for all of the five segments and calculated the mean and standard error of the prediction accuracy on the test set elements.

The tree structure is identical and the software is identical.

SCIENTIFIC REPORTS

Performance comparison between SVM-GKM and kmer-SVM. The k-mer is a widely used feature considering the local sequence order information along the DNA sequences. GKM is an improvement of k-mer by introducing the gaps into k-mers. For comparison, a predictor called kmer-SVM is constructed based on k-mers. The kmer-SVM can be viewed as a special case of GKM-SVM without gaps. Therefore, the implementation of kmer-SVM is the same as that of SVM-GKM except that the gap number n is set as 0, and the tree structure is also employed so as to reduce the computational cost. The performance of these two methods on the benchmark dataset with different parameters is shown in Fig. 2.

As shown in Fig. 2, SVM-GKM consistently outperforms kmer-SVM, especially for lager word length values ($k > 9$). We can also see that parameter k does not have significant impact on the performance of SVM-GKM, and SVM-GKM achieves its highest accuracy (86.57%) when $k = 13$. In contrast, kmer-SVM achieves its highest accuracy (82.31%) when $k = 10$ and then its performance decreases significantly. This is because when k is larger than 10, the dimension of the feature vectors is very large and many values are zeros, leading to extremely sparse problem. For example, when $k = 13$, the dimension of the feature vectors generated by kmer-GKM is

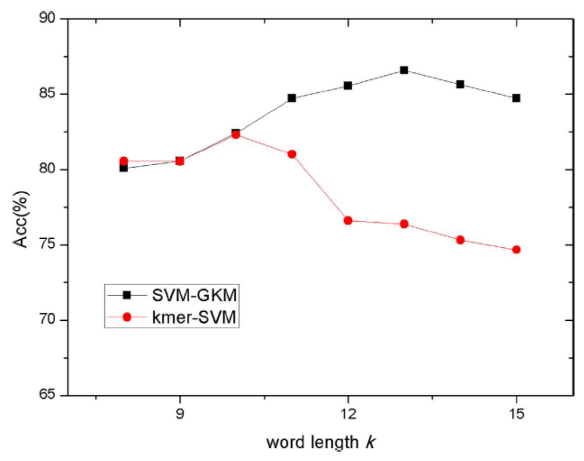


Figure 2. **The influence of parameter k on the performance of two predictors.** Two predictors, one is SVM-GKM, the other is kmer-SVM. We consider the word length k from 8 to 15, and choose the mismatch length $m = 7$ for SVM-GKM predictor. SVM-GKM achieves the highest result when $k = 13$, kmer-SVM obtains the highest result when $k = 10$.

Enhanced Regulatory Sequence Prediction Using Gapped *k*-mer Features

Mahmoud Ghandi[1¶¤], Dongwon Lee[1¶], Morteza Mohammad-Noori[2,3], Michael A. Beer[1,4*]

Gapped *k*-mer SVM classifier outperforms *k*-mer SVM classifier

We compared the performance of gkm-SVM and kmer-SVM on the CTCF data set for a range of oligomer lengths by varying either k (for kmer-SVM) or l (for gkm-SVM) from 6 to 20. We fixed the parameter $k = 6$ for gkm-SVM. We then quantified the classification performance of each by calculating test-set AUC with standard five-fold cross validation (CV) (see Methods). Figure 1A shows a summary of the comparisons. As anticipated, gkm-SVM performs consistently better than kmer-SVM for all lengths. More significantly, while kmer-SVM suffers severely from overfitting when k is greater than 10, gkm-SVM is virtually unaffected by l. In fact, gkm-SVM achieves the best result (AUC = 0.967) when $l = 14$ and $k = 6$, which is significantly better than the kmer-SVM (AUC = 0.912 when $k = 10$); the best ROC curve is shown in

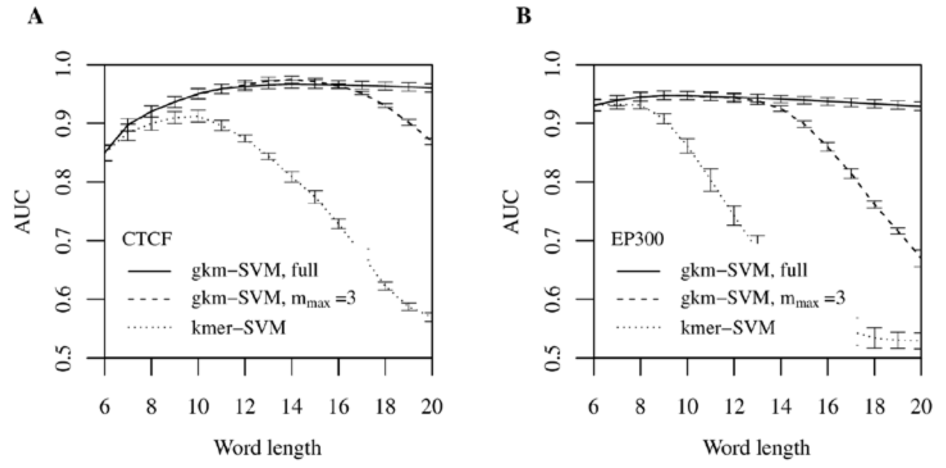


Figure 1. **gkm-SVM outperforms kmer-SVM over a wide range of *k*-mer length.** Both gkm-SVM and kmer-SVM were trained on (A) CTCF bound and (B) EP300 bound genomic regions using different word lengths (k for kmer-SVM and l for gkm-SVM). The parameter k for gkm-SVM was