Detecting Change Points Using Kernel SVMs

Cody Buntain

1 Overview

Offline, parametric methods for change point detection often outperform their online, non-parametric competitors, but the flexibility gained by looser model restrictions and the ability to run in a streaming context are at times more valuable. The kernel-based change detection (KCD) algorithm proposed by Desbory et al. is one such online algorithm [1]. Rather than rely on a priori knowledge of the generating distribution for a given series of data, KCD instead leverages support vector machines (SVMs) to construct descriptions of the data in a higher dimensional space defined by some given kernel (for our cases, the radial basis function, or RBF, kernel). One can then use these high-dimensional descriptions to develop a dissimilarity statistic that characterizes the differences in center and spread of two possible regimes in the data. Additionally, SVM's popular kernel trick allows one to generate this dissimilarity statistic in input space rather than feature space.

1.1 Geometric Interpretation

Desobry et al. provide a compelling geometric interpretation of the KCD based on angles between vectors as a measure of dissimilarity and spread (see Figure 1 for a two-dimensional view). This geometric interpretation exists in feature space and assumes that two one-class SVMs were used to describe data from a possible past regime and a possible future regime respectively. The angle between the vectors normal to the decision planes $\widehat{c_p c_f}$ can then be used as a dissimilarity metric between the two sets of data. To account for within-class spread, the authors then normalize this angle by the sum of the angles between the respective normal vectors a support vector of each regime $\widehat{c_p s_p}$ and $\widehat{c_f s_f}$. The result is the KCD dissimilarity statistic shown in Eq. 1.

$$KCD_{stat} = \frac{\widehat{c_p c_f}}{\widehat{c_f s_f} + \widehat{c_p s_p}} \tag{1}$$

These angles can be calculated using the inverse cosine of the dot product between the weight vectors, as shown in Eq. 2. One should note the denominator in Eq. 2 is for normalizing the vectors to unit length. This normalized dot product can be calculated in input space using SVM's learned weights $\boldsymbol{\alpha_p}$ and $\boldsymbol{\alpha_f}$ and the kernel matrices K as shown in Eq. 3. To obtain the within-class spread for each class, we use a similar form but rely on the SVM's intercept ρ rather than a dot product, as shown in Eq. 4.

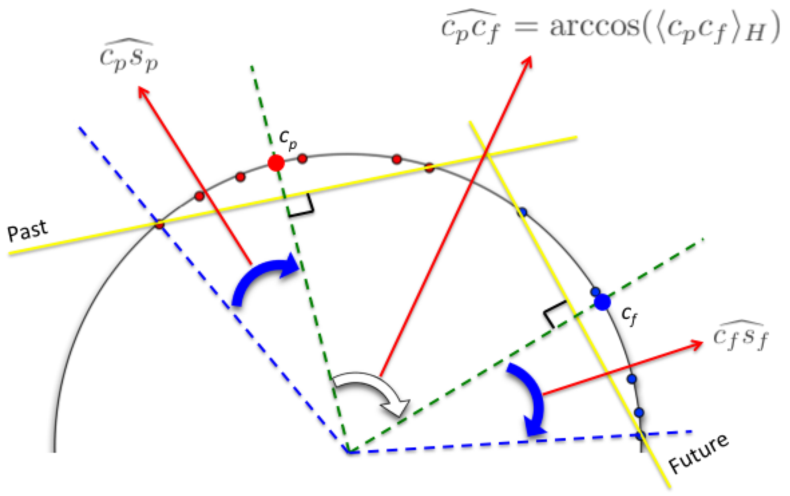

Figure 1: KCD Geometry in Feature Space

$$\widehat{c_p c_f} = \arccos\left(\frac{\langle w_p, w_f \rangle_F}{\|w_p\|\|w_f\|}\right) \tag{2}$$

$$\frac{\langle w_p, w_f \rangle_F}{\|w_p\|\|w_f\|} = \frac{\boldsymbol{\alpha}_p^T K_{p,f} \boldsymbol{\alpha}_f}{\sqrt{\boldsymbol{\alpha}_p^T K_{p,p} \boldsymbol{\alpha}_p}\sqrt{\boldsymbol{\alpha}_f^T K_{f,f} \boldsymbol{\alpha}_f}} \tag{3}$$

$$\widehat{c_i s_i} = \arccos\left(\frac{\rho_i}{\sqrt{\boldsymbol{\alpha}_i^T K_{i,i} \boldsymbol{\alpha}_i}}\right), i \in \{p, f\} \tag{4}$$

2 Algorithm

KCD has four parameters:

- m – Window size, or the number of points on either side of a candidate change point.
- γ – SVM parameter governing bandwidth for the RBF kernel (other kernels are possible, but we did not experiment with them).
- ν – One-class SVM parameter governing proportion of points that should be counted as outliers when training.
- η – Threshold parameter such that, if $KCD_{stat} \geq \eta$ for some time point h, h is considered a change point.

We assume X is the input data of size $n \times k$.

Function KCD(X, m, γ, ν, η) Algorithm by Desobry et al.[1]

```
$n \leftarrow$ len($X$) ;
changePoints $\leftarrow$ [ ] ;
for $h \in [m, n-m-1]$ do
    $X_p \leftarrow X[h-m, h]$ ;
    $X_f \leftarrow X[h, h+m]$ ;
    svm$_p \leftarrow$ SVM.fit($X_p, \gamma, \nu$) ;
    svm$_f \leftarrow$ SVM.fit($X_f, \gamma, \nu$) ;
    $\boldsymbol{\alpha}_p \leftarrow$ svm$_p.\alpha$ ;
    $\boldsymbol{\alpha}_f \leftarrow$ svm$_f.\alpha$ ;
    $\boldsymbol{\rho}_p \leftarrow$ svm$_p.\rho$ ;
    $\boldsymbol{\rho}_f \leftarrow$ svm$_f.\rho$ ;
    $K_{p,p} \leftarrow$ rbf($X_p, X_p$) ;
    $K_{f,f} \leftarrow$ rbf($X_f, X_f$) ;
    $K_{p,f} \leftarrow$ rbf($X_p, X_f$) ;
    $\widehat{c_p c_f} \leftarrow \arccos\left(\frac{\boldsymbol{\alpha}_p^T K_{p,f} \boldsymbol{\alpha}_f}{\sqrt{\boldsymbol{\alpha}_p^T K_{p,p} \boldsymbol{\alpha}_p}\sqrt{\boldsymbol{\alpha}_f^T K_{f,f} \boldsymbol{\alpha}_f}}\right)$ ;
    $\widehat{c_p s_p} \leftarrow \arccos\left(\frac{|\boldsymbol{\rho}_p|}{\sqrt{\boldsymbol{\alpha}_p^T K_{p,p} \boldsymbol{\alpha}_p}}\right)$ ;
    $\widehat{c_f s_f} \leftarrow \arccos\left(\frac{|\boldsymbol{\rho}_f|}{\sqrt{\boldsymbol{\alpha}_f^T K_{f,f} \boldsymbol{\alpha}_f}}\right)$ ;
    $KCD_{stat} \leftarrow \frac{c_p c_f}{c_p s_p + c_f s_f}$ ;
    if $KCD_{stat} > \eta$ then
        changePoints += $h$
    end
end
return changePoints
```

References

[1] F. Desobry, M. Davy, and C. Doncarli. An online kernel change detection algorithm. *Signal Processing, IEEE Transactions on*, 53(8):2961–2974, Aug. 2005.