Multiple Locally Linear Kernel Machines

David Picard   DAVID.PICARD@ENPC.FR

LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-Vallée, France

Abstract

In this paper we propose a new non-linear classifier based on a combination of locally linear classifiers. A well known optimization formulation is given as we cast the problem in a ℓ_1 Multiple Kernel Learning (MKL) problem using many locally linear kernels. Since the number of such kernels is huge, we provide a scalable generic MKL training algorithm handling streaming kernels. With respect to the inference time, the resulting classifier fits the gap between high accuracy but slow non-linear classifiers (such as classical MKL) and fast but low accuracy linear classifiers.

1. Introduction

SVMs were shown to provide very accurate classifiers, and are consequently used in a growing number of applications. SVM training algorithms and subsequent inference efficiency can be separated in two groups, depending on the kernel being linear or not. When dealing with linear SVMs, very efficient training algorithms based on stochastic gradient descent such as (Shalev-Shwartz et al., 2007; Bordes et al., 2009; Roux et al., 2012) allow to deal with large datasets in a short amount of time. The inference time of linear SVMs is clearly their main advantage as it can efficiently be optimized for modern computer architecture. However, many classification problems are not linearly separable and consequently a non linear kernel is required.

Kernel SVMs also benefit from recent developments in stochastic coordinate methods such as (Bordes et al., 2005; Shalev-Shwartz & Zhang, 2013; 2014) and come now with efficient training procedures. The main drawback of Kernel SVMs is the inference time which is proportional to the number of support vectors. This number has been experimentally shown to grow linearly with the size of the

---

training set (Bordes et al., 2005). Most of the speed-up is obtained by limiting the number of support vector (Dekel & Singer, 2006). For example, in (Ertekin et al., 2011), the authors proposed to ignore outliers to reduce the number of support vectors, but the problem becomes non-convex and the resulting classifier still has a high inference time.

With respect to the inference cost, a much more efficient alternative is the use of almost linear classifiers such as Locally Linear SVM (LLSVM) proposed in (Ladicky & Torr, 2011). In LLSVM, a generative manifold learning algorithm produces a partition the input space. An almost piece-wise linear classifier concatenating linear classifier of the different parts is then trained. The inference cost of LLSVM is almost as fast as for linear SVMs, depending only on the number of anchor points used in the manifold learning. However, LLSVMs are dependent on the success of the manifold learning algorithm that provides the anchor points. Moreover, the parameter tuning of such algorithms, mainly the number of anchor points and the coding function, is often costly and difficult to perform.

In this paper, we propose a new locally linear classifier similar to LLSVM, but without the burden of the manifold learning part while keeping very efficient inference procedure. We first define a family of locally linear kernels which are related to conformal kernels (Amari & Wu, 1999). We then used the Multiple Kernel Learning (MKL) framework (Bach et al., 2004) to select a subset of the locally linear kernels. The ℓ_1-norm constraint on the kernel combination leads to a limited number of selected kernels and consequently leads to low inference cost. Our contributions can be summed up as the following: 1) we propose a new learning problem named Multiple Locally Linear Kernel Machine (MLLKM) to obtain locally linear classifiers more easily. 2) Since MLLKM is similar to ℓ_1-MKL, we propose a new ℓ_1-MKL solver that can handle the high number of kernels in MLLKM.

The remaining of this paper is organized as follows. In the next section, we review existing works on locally linear classifiers. In Section 3, we detail the concept of locally linear kernels. Then, we present our proposed MLLKM problem and show it is equivalent to solving ℓ_1-MKL. In the same section, we present a fast algorithm to solve ℓ_1-

MKL problems on a budget and show several strategies to automatically tune the parameters and drastically lower the number of selected kernels. We present experiments in Section 5 before we conclude.

2. Locally Linear Classifiers

Generally, a locality criterion in machine learning refers to an adaptation of the model with respect to the localization in the input space. For instance, the idea of locally linear classification is to consider a combination of different linear predictors depending on some locality criterion performed on the evaluated sample. Remark that although the predictors are linear, the combination might not, and consequently the resulting classifier can perform non-linear separation.

In Locally Linear SVM (Ladicky & Torr, 2011), a dictionary learning algorithm is used to provide a set of anchor points $\{\mathbf{x}_c\}$ that describe the manifold of the data. This set is usually obtained using a k-means clustering of a large set of (unlabeled) samples, although more sophisticated method can be used. Any sample $\mathbf{x}$ can then be coded by the anchor points $\mathbf{x}_c$ using local coordinate $\gamma_{\mathbf{x}_c}(\mathbf{x})$ such that $\sum_{\mathbf{x}_c} \gamma_{\mathbf{x}_c}(\mathbf{x}) = 1$ and $\mathbf{x} \approx \sum_{\mathbf{x}_c} \gamma_{\mathbf{x}_c}(\mathbf{x})\mathbf{x}_c$ (Yu et al., 2009). Examples of such local coordinate coding include Kernel Codebook (van Gemert et al., 2008) and Locality constraint Linear Coding (Wang et al., 2010). These local coordinate are then used to perform the combination of local classifiers:

$$f(\mathbf{x}) = \gamma_{\mathbf{x}_c}(\mathbf{x})^\top \mathbf{W}\mathbf{x} + \gamma_{\mathbf{x}_c}(\mathbf{x})^\top \mathbf{b},$$

with $\mathbf{W}$ being the matrix which lines are the local hyperplanes and $\mathbf{b}$ is a vector of local biases. To train $\mathbf{W}$ and $\mathbf{b}$, very efficient stochastic gradient descent algorithms can be used on each local hyperplane weighted by the local coordinate. Extension to multiclass problems can be achieve by replacing the hinge loss with a multiclass loss as presented in (Fornoni et al., 2013).

In (Gönen & Alpaydin, 2008), the authors proposed to tackle the localization as a multiple kernel learning problem, using quasi-conformal transformation:

$$K_\eta(\mathbf{x}_i, \mathbf{x}_j) = \sum_m \eta_m(\mathbf{x}_i)\eta_m(\mathbf{x}_j)K_m(\mathbf{x}_i, \mathbf{x}_j),$$

where $\eta_m(\cdot)$ is a locality function that aims at selecting the right kernel among the K_m depending on $\mathbf{x}_i$ and $\mathbf{x}_j$. As in LLSVM, the authors proposed each locality function η_m to be associated with an anchor point $\mathbf{v}_m$:

$$\eta_m(\mathbf{x}) = \frac{exp(\langle \mathbf{v}_m, \mathbf{x} \rangle + v_{m0})}{\sum_k exp(\langle \mathbf{v}_k, \mathbf{x} \rangle + v_{ik0})}$$

The resulting MKL problem has then 2 sets of variables, namely the dual SVM variables α associated with the training samples, and the anchor points $\mathbf{v}_m$ that define the kernels in the combination. The LMKL algorithm consists in an alternate optimization scheme between α and $\mathbf{v}_m$, where the $\mathbf{v}_m$ are optimized using a gradient descent strategy. In case of linear kernels, it is easy to see LLSVM and LMKL lead to the same family of classifiers, with LMKL allowing to tune the anchor points to the specific classification task. However, since the LMKL objective function is non-convex, a local optimum is found.

Both methods rely on a manifold learning procedure to determine the anchor points. The main problem of such method is that it often leads to a non-convex problem for which no guarantee on the solution can be given, and that is likely to have a high computational cost. Moreover, the parameter tuning of such solution (*i.e.*, the number of anchor points, the coding functions) is very difficult to perform. In practice, it relies on an exploration of the parameters space using cross-validation which is very costly.

In this paper, we thus propose to remove the burden of the manifold learning part by casting the selection of the anchor points and the parameters in a MKL problem. To be able to do that, we first define a family of locally linear kernels in the next section.

3. Locally linear kernels

Let $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$ be two element of $\mathbb{R}^d$, and $\langle \mathbf{x}_1, \mathbf{x}_2 \rangle = \mathbf{x}_1^\top \mathbf{x}_2$ the standard dot product . Let the norm $\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$ and the metric $d(\mathbf{x}_1, \mathbf{x}_2) = \|\mathbf{x}_1 - \mathbf{x}_2\|$ be the standard (euclidean) norm and metric associated with $\langle \cdot, \cdot \rangle$.

We propose to explicitly build a Hilbert space $\mathcal{X}_c$ as a subspace of the input space $\mathbb{R}^d$ locally defined around an arbitrarily chosen center $\mathbf{x}_c \in \mathbb{R}^d$. To enforce locality, we center the data on $\mathbf{x}_c$ and then apply a norm scaling function that tends to map the vectors to 0 when the norm becomes to large. The mapping from $\mathbb{R}^d$ to $\mathcal{X}_c$ has the following expression:

$$\begin{aligned} \phi_c : \mathbb{R}^d &\to \mathcal{X} \\ \mathbf{x} &\mapsto h^{(c)}(\mathbf{x})(\mathbf{x} - \mathbf{x}_c) \end{aligned} \tag{1}$$

Where $h^{(c)}$ is a conformal map rendering the vicinity of the given center $\mathbf{x}_c$. We show in Table 1 several examples of such mappings. By definition, $\mathcal{X}_c$ is a Hilbert space endowed with the following explicit dot product $k_c(\cdot, \cdot)$:

$$\begin{aligned} k_c(\mathbf{x}_1, \mathbf{x}_2) &= \langle \phi_c(\mathbf{x}_1), \phi_c(\mathbf{x}_2) \rangle_{\mathcal{X}_c} \\ &= \left(h^{(c)}(\mathbf{x}_1)(\mathbf{x}_1 - \mathbf{x}_c)\right)^\top \left(h^{(c)}(\mathbf{x}_2)(\mathbf{x}_2 - \mathbf{x}_c)\right) \end{aligned}$$

The properties of $\mathcal{X}_c$ depends on the family of mappings h chosen, which we consider to be a hyperparameter, and
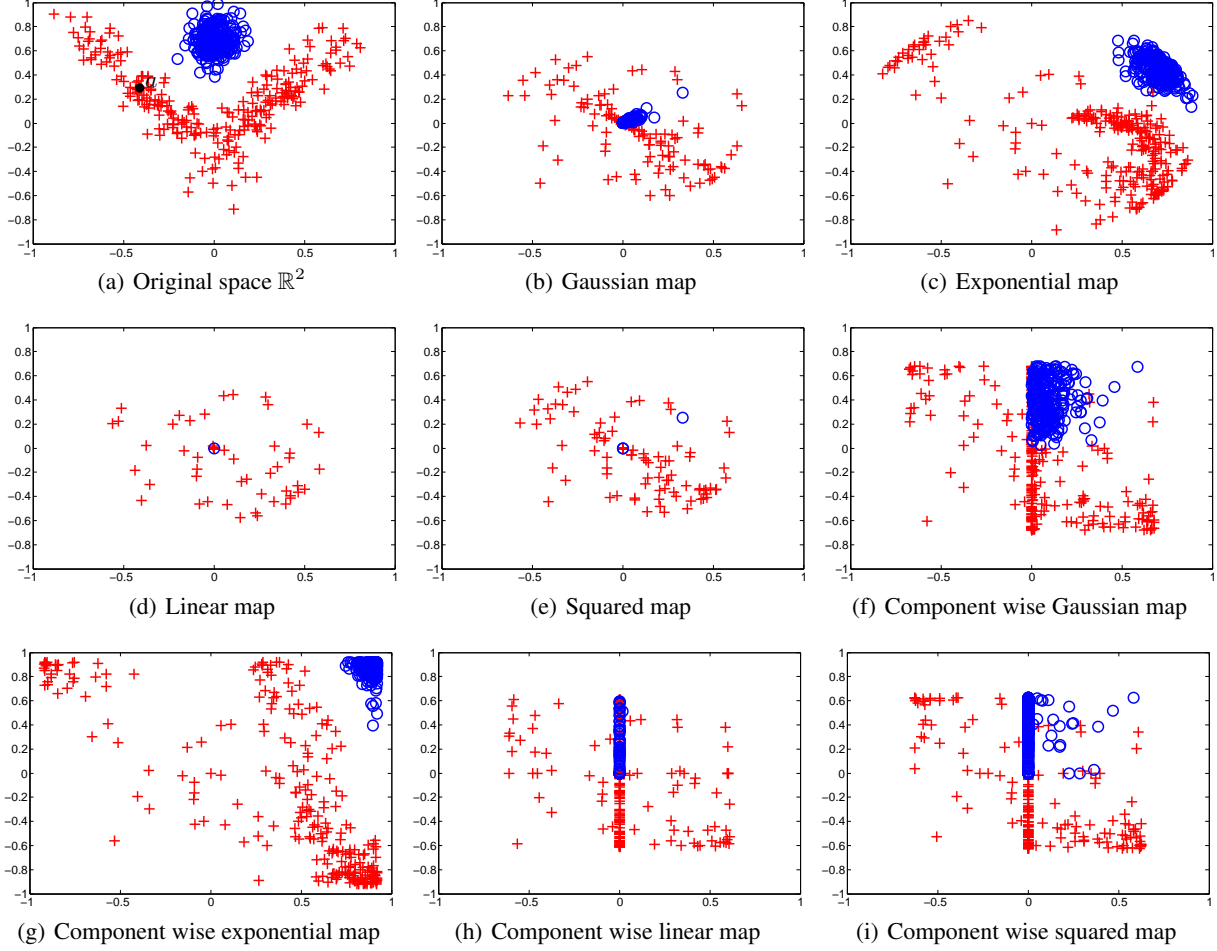
Figure 1. Example of synthetic data and corresponding $\mathcal{X}_c$ for different types of maps, with anchor point a.

Table 1. Examples of conformal maps used in this work, with $|t|_+ = \max(0, t)$.

name	definition	bounded	smooth
$h_e^{(c)}(\mathbf{x})$	$e^{-\gamma\|\mathbf{x}-\mathbf{x}_c\|}$	-	-
$h_g^{(c)}(\mathbf{x})$	$e^{-\gamma\|\mathbf{x}-\mathbf{x}_c\|^2}$	-	✓
$h_l^{(c)}(\mathbf{x})$	$\left\|1 - \gamma\|\mathbf{x}-\mathbf{x}_c\|\right\|_+$	✓	-
$h_s^{(c)}(\mathbf{x})$	$\left\|1 - \gamma\|\mathbf{x}-\mathbf{x}_c\|^2\right\|_+$	✓	✓

its subsequent parameters, namely $\{\gamma, \mathbf{x}_c\}$ in our examples. We show in Figure 1 examples of such mapping on synthetic data. As we can see, many samples are mapped to or close to 0. The effect is all the more visible using bounded maps, since samples outside of the support of ϕ_c are mapped exactly to 0. However, the local geometry of the point cloud is preserved.

To allow for more sophisticated manipulations, we also consider component wise local mappings, in the form of:

$$\phi_c : \mathbb{R}^d \to \mathcal{X} \qquad (2)$$
$$\mathbf{x} \mapsto \mathbf{h}^{(c)}(\mathbf{x}) \circ (\mathbf{x} - \mathbf{x}_c)$$

where $\circ$ is the entry wise (Hadamard) product. With a slight abuse of notations, the mapping $\mathbf{h} : \mathbb{R}^d \mapsto \mathbb{R}^d$ now produces a vector rendering the vicinity of each component to the anchor point. The explicit dot product corresponding to $\mathcal{X}_c$ is then the following:

$$k_c(\mathbf{x}_1, \mathbf{x}_2) =$$
$$\left(h^{(c)}(\mathbf{x}_1) \circ (\mathbf{x}_1 - \mathbf{x}_c)\right)^\top \left(h^{(c)}(\mathbf{x}_2) \circ (\mathbf{x}_2 - \mathbf{x}_c)\right)$$

Using the examples given in Table 1 on each component of input samples $\mathbf{x}$ produces the corresponding component wise mappings. These mappings are also illustrated in Figure 1.

The main difference between the global and the component wise mappings is that the latter makes the assumption

the components can be considered independently, which is the case if the features have been decorrelated (*e.g.*, after a Karhunen-Loève transform).

It is clear that using only one of such kernels (either global or component wise) has less discriminatory capabilities than the simple linear kernel. Indeed, since many features are mapped to 0 independently of their class, it is very likely that the classification problem becomes non linearly separable after the transform. We thus consider the case where several locally linear kernels are used and summed into a single kernel:

$$k(\mathbf{x}_1, \mathbf{x}_2) = \sum_c k_c(\mathbf{x}_1, \mathbf{x}_2) \qquad (3)$$

The resulting space $\mathcal{X}$ is the concatenation of the $\mathcal{X}_c$. Problems that were not linearly separable in $\mathbb{R}^d$ (for example the synthetic data of Figure 1) can be in $\mathcal{X}$, provided a sufficient number of locally linear kernels with the right range of locality are used.

To choose these kernels, we consider the training set $\mathcal{A}$ of samples $\{\mathbf{x}_i\}_i$ of $\mathbb{R}^d$ to be used to train the classifier. We propose to perform a linear combination consisting of a locally linear kernel for each sample:

$$k(\mathbf{x}_1, \mathbf{x}_2) = \sum_{\mathbf{x}_c \in \mathcal{A}} \beta_c k_c(\mathbf{x}_1, \mathbf{x}_2) \qquad (4)$$

We are now left with the optimization of β so as to discard irrelevant kernels from the combination. The next section devises the resulting optimization problem and proposes an algorithm to solve it.

4. Multiple Locally Linear Kernel Machines

Let $\mathcal{A} = \{(\mathbf{x}_i, y_i)\}_{1 \leq i \leq n}$ be a training set of n training samples $\mathbf{x}_i \in \mathbb{R}^d$ and their associated labels $y_i \in \{-1, 1\}$. The Multiple Locally Linear Kernel Machine (MLLKM) is then the optimal predictor according to the following primal problem:

$$\min_{w, \beta, \zeta} \quad \frac{1}{2} \sum_{m=1}^n \|w_m\|_{\mathcal{H}_m}^2 + C \sum_{i=1}^n \zeta_i \qquad (5)$$

$$s.t. \quad \forall i, y_i \sum_m \sqrt{\beta_m} w_m^\top \phi_m(\mathbf{x}_i) \geq 1 - \zeta_i$$

$$\forall i, \zeta_i \geq 0$$
$$\forall m, \beta_m \geq 0$$
$$\sum_m \beta_m = 1$$

Remark this is corresponds to a standard ℓ_1-constraint MKL problem with the slight difference that the number of kernels is the same as the number of training samples.

On many occasions, ℓ_p-MKL with $p > 1$ are known to perform better than ℓ_1 constraint (Kloft et al., 2009), however the number of kernels in MLLKM is too high to allow for non sparse combinations. In such case, the computational benefit of having a locally linear classifier would be lost to the very high number of linear predictors in the resulting combination.

To recover classical formulation of ℓ_1-MKL, we consider the following Lagrangian of the primal problem with β left in the constraints:

$$\mathcal{L}(w, \beta, \zeta, \alpha, \lambda) = \frac{1}{2} \sum_m \|w_m\|^2 + \sum_i (C - \lambda_i - \alpha_i) \zeta_i$$

$$- \sum_i \alpha_i \left[y_i \sum_m \sqrt{\beta_m} w_m^\top \phi_m(\mathbf{x}_i) - 1 \right]$$

$$s.t. \quad \forall m, \beta_m \geq 0$$
$$\sum_m \beta_m = 1$$

The KKT stationary condition on w states that:

$$\frac{\partial \mathcal{L}}{\partial w_m^\star} = 0 = w_m^\star - \sum_i \alpha_i y_i \sqrt{\beta_m^\star} \phi_m(\mathbf{x}_i)$$

Or equivalently:

$$w_m^\star = \sum_i \alpha_i y_i \sqrt{\beta_m^\star} \phi_m(\mathbf{x}_i)$$

Similarly, the stationary condition on ζ states that:

$$\frac{\partial \mathcal{L}}{\partial \zeta_i} = 0 = C - \alpha_i - \lambda_i$$

Since α and λ are dual variables, the dual feasibility condition imposes their positiveness, and in particular implies a box constraint on α:

$$\forall i, 0 \leq \alpha_i \leq C$$

A dual formulation of problem (5) for w and ζ variables is then obtained by injecting these conditions into the original formulation:

$$\mathcal{D}(\alpha, \beta) = \inf_{w, \zeta} \mathcal{L}(w, \beta, \zeta, \alpha, \lambda)$$

$$= \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \sum_m \beta_m k_m(\mathbf{x}_i, \mathbf{x}_j)$$

$$s.t. \quad \forall m, \beta_m \geq 0$$
$$\sum_m \beta_m = 1$$

This dual expression allows us to recover to the well known $\min \max$ formulation used for example in (Rakotoma-

monjy et al., 2008):

$$\min_\beta \max_\alpha \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \sum_m \beta_m k_m(\mathbf{x}_i, \mathbf{x}_j)$$

$$(6)$$

$$s.t. \quad \forall i, 0 \leq \alpha_i \leq C$$
$$\forall m, \beta_m \geq 0$$
$$\sum_m \beta_m = 1$$

Since the number of kernels is equal to the number of training samples, large datasets lead to optimization problem intractable for current algorithms. In particular, most algorithms suppose all kernel matrices can be computed and stored in memory beforehand, which is clearly not the case when considering even medium sized datasets (*i.e.*, more than 10k samples). Consequently, we present a new algorithm called *SequentialMKL*, for solving large MKL problems with reduced memory footprint that is suitable for training MLLKM.

4.1. Sequential MKL

The main idea in SequentialMKL is to consider a reduced active set of kernels, solve the MKL problem for this reduced set, and then probe new kernels for inclusion in the set. Solving the MKL problem with a reduced set of active kernels can be done efficiently using existing solver like (Sonnenburg et al., 2006; Chapelle & Rakotomamonjy, 2008) or (Rakotomamonjy et al., 2008). In our case, we choose the reduced gradient approach of SimpleMKL of (Rakotomamonjy et al., 2008) since it is closely related to the inclusion criterion of new kernels. For the internal SVM solver, we use a variant of SDCA (Shalev-Shwartz & Zhang, 2013) presented in Algorithm 1. We keep track of the outputs $\hat{\mathbf{y}}$ of the classifier and check for an early bail out criterion to improve the cost of each iteration. To design the criterion for inserting new kernels to the active set, we consider a Lagrangian of the $\min\max$ problem 6 where the constraints of β are taken into account:

$$\mathcal{M}(\alpha, \beta, \mu, \nu) = \sum_i \alpha_i - \sum_m \mu_m \beta_m - \nu\left(\sum_m \beta_m - 1\right)$$
$$- \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \sum_m \beta_m k_m(\mathbf{x}_i, \mathbf{x}_j)$$

Now, the stationary condition at optimum $\beta^\star$ imposes that

$$\nu^\star = -\mu_m^\star - \frac{1}{2}\sum_{i,j} \alpha_i^\star \alpha_j^\star y_i y_j k_m(\mathbf{x}_i, \mathbf{x}_j)$$

Algorithm 1 SDCA for solving SVM

  function SDCA($\mathbf{y}, K, E, C$)
    $\hat{\mathbf{y}} \leftarrow \mathbf{0}, \alpha \leftarrow \mathbf{0}, e \leftarrow 0$
    repeat
      $L \leftarrow$ random permutation of $\{1, ..., n\}$
      for $i \in L$ **do**
        $g \leftarrow 1 - \mathbf{y}_i \hat{\mathbf{y}}_i$
        if $g = 0$ **or** ($g > 0$ **and** $\alpha_i = C$)
                  or ($g < 0$ **and** $\alpha_i = 0$)
        then
          skip i
        end if
        $\alpha_{new} \leftarrow \max(0, \min(\alpha_i + g/K_{ii}, C))$
        $\delta \leftarrow \alpha_{new} - \alpha_i$
        for $j \in \{i, ..., n\}$ **do**
          $\hat{\mathbf{y}}_j \leftarrow \hat{\mathbf{y}}_j + \delta \mathbf{y}_j K_{ij}$
        end for
      end for
      $e \leftarrow e + 1$
    until $e \geq E$
    return α
  end function

Or, equivalently $\forall m, \forall l$:

$$\mu_m^\star + \frac{1}{2}\sum_{i,j}\alpha_i^\star \alpha_j^\star y_i y_j k_m(\mathbf{x}_i, \mathbf{x}_j) =$$

$$\mu_l^\star + \frac{1}{2}\sum_{i,j}\alpha_i^\star \alpha_j^\star y_i y_j k_l(\mathbf{x}_i, \mathbf{x}_j)$$

Let us denote k_k a kernel with non-zero weight $\beta_k^\star$ at optimum, then $\mu_k^\star = 0$ by complementary slackness. In turns, it means that $\forall l \neq k$,

$$\frac{1}{2}\sum_{i,j}\alpha_i^\star \alpha_j^\star y_i y_j k_k(\mathbf{x}_i, \mathbf{x}_j) = \mu_l + \frac{1}{2}\sum_{i,j}\alpha_i^\star \alpha_j^\star y_i y_j k_l(\mathbf{x}_i, \mathbf{x}_j)$$

$$\geq \frac{1}{2}\sum_{i,j}\alpha_i^\star \alpha_j^\star y_i y_j k_l(\mathbf{x}_i, \mathbf{x}_j)$$

$$(7)$$

since by dual feasibility all $\mu_l^\star \geq 0$. The criterion thus consists in finding a kernel of the open kernel set violating constraint (7) and adding it to the current active kernel set. For numerical stability reasons, we compute the largest $\frac{1}{2}\sum_{i,j}\alpha_i \alpha_j y_i y_j k_m(\mathbf{x}_i, \mathbf{x}_j)$ among non zero weight kernels k_m. Remark that kernels respecting Equation (7) have zero weight and μ_m equals to the difference of gradients with k_k, meaning the KKT conditions are respected.

The full SequentialMKL algorithm is presented in Algorithm 2. We alternate between optimizing the kernel weights and search for new kernel to insert in the active set, until no new kernel can be inserted. In practice, we
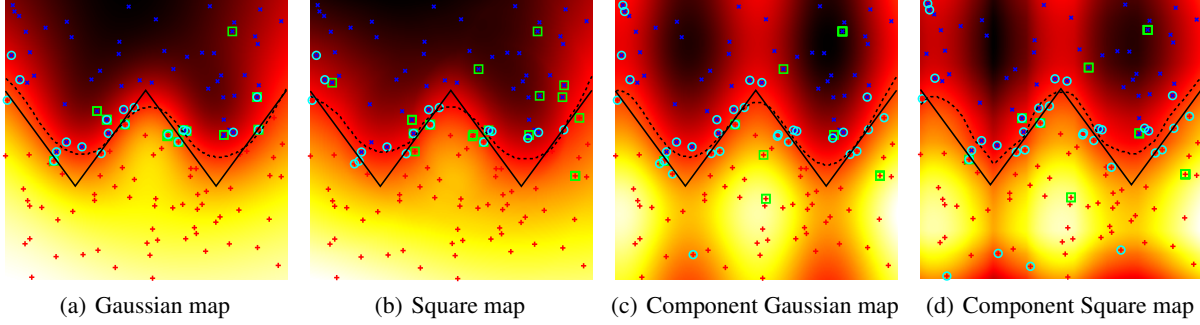
(a) Gaussian map   (b) Square map   (c) Component Gaussian map   (d) Component Square map

Figure 2. Example of synthetic data classification task. The true boundary is shown in plain black line, while the classifier boundary is in dashed black line. The circles denote the support vectors, while the squares denote the support kernels.

Algorithm 2 SequentialMKL

> **function** TRAIN($\{k_m\}, \mathbf{x}, \mathbf{y}, E, C$)
> $\quad \mathcal{S} \leftarrow k_1, \mathcal{O} \leftarrow \{k_m\}\backslash k_1$
> $\quad \beta \leftarrow \mathbf{1}, \alpha \leftarrow \mathbf{0}$
> $\quad$ **repeat**
> $\quad\quad K \leftarrow \sum_{k_m \in \mathcal{S}} \beta_m k_m$
> $\quad\quad \alpha \leftarrow \text{SDCA}(\mathbf{y}, K, E, C)$
> $\quad\quad term \leftarrow \mathbf{true}$
> $\quad\quad g \leftarrow \max_{k_m \in \mathcal{S}} \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k_m(i,j)$
> $\quad\quad$ **for** $k_m \in \mathcal{O}$ **do**
> $\quad\quad\quad$ **if** $\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k_m(i,j) > g$ **then**
> $\quad\quad\quad\quad term \leftarrow \mathbf{false}$
> $\quad\quad\quad\quad \mathcal{S} \leftarrow \mathcal{S} + k_m, \mathcal{O} \leftarrow \mathcal{O}\backslash k_m$
> $\quad\quad\quad$ **end if**
> $\quad\quad$ **end for**
> $\quad\quad \beta \leftarrow \text{SolveMKL}(\mathcal{S}, \beta, \mathbf{x}, \mathbf{y}, E, C)$
> $\quad\quad$ **for** $\beta_m \in \beta$ **do**
> $\quad\quad\quad$ **if** $\beta_m = 0$ **then**
> $\quad\quad\quad\quad \mathcal{S} \leftarrow \mathcal{S}\backslash k_m, \mathcal{O} \leftarrow \mathcal{O} + k_m$
> $\quad\quad\quad$ **end if**
> $\quad\quad$ **end for**
> $\quad$ **until** $term$
> $\quad$ **return** α, β
> **end function**

---

found that inserting more than one kernel at a time do not increase the time taken to update the weights, while significantly reducing the number of iterations of the outer loop. Indeed, there is a tradeoff between inserting many kernels and performing several passes over the entire kernel set that can be roughly decided by the maximum number of Gram matrices that can be cached in memory to allow fast training in the inner loop.

Moreover, SequentialMKL can very simply be extended to an online variant where the kernels are streamed. In such case, each time a new kernel is considered, it is probed for insertion using the gradient criterion, and all weights are

consequently re-optimized (both α and β). To obtain the true MKL solution, all previously discarded kernels have to be probed after each new insertion, which can be done using a reprocess function akin to the one in LaSVM (Bordes et al., 2005). When working on a budget, discarded kernels can be forgotten, but then the solution given by SequentialMKL is only approximated unless many passes over the whole kernel set are made. In the specific case of MLLKM, a fully online algorithm can be efficiently designed, where training samples are streamed and the corresponding kernels are probed at the same time.

4.2. Inference computational cost

Like in many MKL problem, the trained classification function is the following:

$$f(x) = \sum_i \alpha_i \sum_c \beta_c k_c(x_i, x). \tag{8}$$

For the quasi-conformal mapping, the expression of f can be further developed using the explicit feature space:

$$f(x) = \sum_i \alpha_i \sum_c \beta_c h_c(x_i) h_c(x) (x_i - x_c)^\top (x - x_c). \tag{9}$$

Most of its computation can then be accelerated using this explicit mapping:

$$f(x) = \sum_c h_c(x)(x - x_c)^\top w_c, \tag{10}$$

with

$$w_c = \beta_c \sum_i \alpha_i h_c(x_i)(x_i - x_c). \tag{11}$$

The same can be done for the component wise mappings.

Using this form, the inference cost is expected to be low since the algorithm is set to have a sparsity pattern on β due to the ℓ_1 norm constraint. Moreover, the evaluation in this form independent of the number of support vectors, just as for linear SVM. Remark that this inference cost is

valid both in terms of number of operations and in memory requirement. In case where the classifier has to fit a low memory hardware at inference time, MLLKM has clearly the advantage that only a few number of parameters have to be stored compared with kernel SVM trained on a huge training set.

4.3. Parameter selection

For the mapping function of Table 1, the γ parameters have to be tuned to take into account the scale of the data. To avoid the burden of doing so by cross-validation, a sampling over a reasonable segment (*e.g.*, $[0.01, 10]$) can be made and all the corresponding locally linear kernels related to the training samples are considered in the initial kernel set. Although it raises dramatically the total number of kernels, we found this solution to be more efficient in training time than choosing the right parameter using cross-validation. Furthermore, it allows for different parameters for different kernels, which betters renders the different scales of the data.

5. Experiments

We test our algorithms on 2 different datasets. The first dataset is a synthetic binary classification task with highly non-linear boundary between the classes. Then we compare our algorithm performances on well known datasets of the UCI repository.

5.1. Synthetic data

We first present experiments on a synthetic dataset containing two uniformly distributed classes, separated by a piecewise linear boundary. Figure 2 shows the output values of the classifier as a heat map, the true boundary in continuous line, the boundary of the classifier in dashed line, support vectors in circles and active kernels in squares. C was set to 1000.

As we can see, we can achieve a curved boundary thanks to the selected locally linear kernels. In particular, the square and component-wise square maps have higher curvature (infinite in the case of component wise mapping), which may be due to the bounded support of such maps. The component wise maps have less active kernels, although they provide a comparable boundary. Remark the support vectors are only shown to measure the extend of the support domain for the boundary, since they are not used in the computation of the output. Using conformal map leads to support vectors concentrated around the boundary, while using component siwe maps leads to support vectors scattered inside the maximum ℓ_∞ bounding box the the active kernel set.

5.2. UCI dataset

We evaluated our MLLKM on several UCI datasets (ionoshpere, sonar, heart and diabetes), following the procedure of (Rakotomamonjy et al., 2008) which is the most used in MKL evaluations. For fairness of evaluation with respect to the computational time, we implemented both MLLKM and SequentialMKL using the JKernelMachines library (Picard et al., 2013) which already contains several algorithms against which we compared our work, *i.e.*, LaSVM (Bordes et al., 2005), SAG (Roux et al., 2012), LLSVM (Ladicky & Torr, 2011) and SimpleMKL (Rakotomamonjy et al., 2008). We performed 10 random split of the data where 70% was kept for training, and measured the average accuracy. C was arbitrarily set to 100 for all algorithms and all datasets.

When using MKL, we used the standard setup that consists in adding Gaussian kernels for each component (10 bandwidths) as well as homogeneous and inhomogeneous polynomial kernels of degrees up to 3. The total number of kernels in such case is 19 times the dimension of the input space. For LLSVM, we arbitrarily set the number of anchor points to 32, while the bandwith of the Gaussian kernel for LaSVM was set using crossvalidation.

We show in Table 2 the accuracy, the mean inference time and the number of active kernels and support vectors for all methods. The results are grouped into 3 categories: Our method with varying kernel types, linear or locally linear methods and non-linear methods. As we can see MLLKM is in general close to non-linear methods in terms of accuracy, while having a much reduced inference time. Except for the heart dataset, MLLKM always outperforms linear methods by a fair margin while having a comparable inference time.

One interesting property of MLLKM is with respect to the storage cost of the classifier, which is proportional to the number of selected kernels. On the contrary, Kernel SVM and MKL have to store the support vectors. As we can see, the number of selected kernels in MLLKM is much lower than the number of support vector in Kernel SVM and MKL (by a factor of 5 to 10 in case of MKL), which means the resulting classifier is easier to embed in a low memory device even when trained on a large dataset.

6. Conclusion

In this paper, we presented a new classifier named Multiple Locally Linear Kernel Machine based on the combination of locally linear kernel. The proposed approach has several advantages: First it fits perfectly between high accuracy kernel SVM that have a high inference cost and lower accuracy linear SVM with a low inference cost. Second, the proposed problem is equivalent to ℓ_1-MKL, which

Table 2. Results on several UCI datasets. Methods are grouped into categories representing MLLKM with varying kernel type, linear and locally linear methods and non-linear methods.

METHOD	ACCURACY	INFERENCE TIME (MS)	#KERNELS× #SUPPORT VECTORS
IONOSPHERE			
MLLKM (GAUSSIAN)	**94.0** ± 2.3	**1.0** ± 0.0	22.5 ± 7.9 × 1
MLLKM (SQUARE)	92.9 ± 2.8	2.4 ± 3.0	25.9 ± 9.0 × 1
MLLKM (COMPONENT GAUSSIAN)	91.7 ± 1.1	8.0 ± 2.2	27.0 ± 5.7 × 1
MLLKM (COMPONENT SQUARE)	91.4 ± 2.4	1.9 ± 0.8	25.3 ± 7.4 × 1
SAG	79.1 ± 4.3	0.0 ± 0.0	1 × 1
LLSVM	**87.3** ± 2.2	**1.5** ± 1.4	32 × 1
LaSVM (GAUSSIAN)	94.3 ± 2.2	1.2 ± 0.6	1 × 53.3 ± 4.3
SIMPLEMKL	**95.1** ± 2.2	**18.9** ± 8.1	9.5 ± 4.4 × 159.7 ± 13.1
SEQUENTIALMKL	93.3 ± 3.1	55.2 ± 22.5	15.9 ± 8.1 × 141.8 ± 30.9
SONAR			
MLLKM (GAUSSIAN)	**81.0** ± 2.8	**0.6** ± 0.5	17.6 ± 5.5 × 1
MLLKM (SQUARE)	80.2 ± 3.5	0.8 ± 0.4	18.4 ± 5.4 × 1
MLLKM (COMPONENT GAUSSIAN)	77.8 ± 2.4	4.9 ± 1.8	19.8 ± 7.3 × 1
MLLKM (COMPONENT SQUARE)	80.2 ± 3.0	1.0 ± 0.4	23.1 ± 5.0 × 1
SAG	62.5 ± 5.9	0.0 ± 0.0	1 × 1
LLSVM	**67.8** ± 5.4	**0.5** ± 0.5	32 × 1
LaSVM (GAUSSIAN)	79.0 ± 3.5	0.5 ± 0.5	1 × 73.6 ± 4.1
SIMPLEMKL	84.6 ± 4.4	10.7 ± 1.9	8.5 ± 1.1 × 138.1 ± 2.9
SEQUENTIALMKL	**86.3** ± 2.7	**15.4** ± 12.0	11.2 ± 9.2 × 124.5 ± 26.2
HEART			
MLLKM (GAUSSIAN)	81.6 ± 3.0	0.6 ± 0.5	26.0 ± 8.3 × 1
MLLKM (SQUARE)	81.4 ± 4.1	0.5 ± 0.5	27.1 ± 11.6 × 1
MLLKM (COMPONENT GAUSSIAN)	**82.2** ± 4.9	**2.2** ± 0.4	27.3 ± 6.3 × 1
MLLKM (COMPONENT SQUARE)	81.1 ± 2.8	0.8 ± 0.6	26.8 ± 6.9 × 1
SAG	**83.0** ± 3.5	**0.1** ± 0.3	1 × 1
LLSVM	80.4 ± 4.1	0.3 ± 0.5	32 × 1
LaSVM (GAUSSIAN)	79.6 ± 2.4	0.5 ± 0.5	1 × 74.7 ± 5.2
SIMPLEMKL	**80.2** ± 2.9	**10.6** ± 2.2	7.4 ± 0.9 × 163.2 ± 5.7
SEQUENTIALMKL	80.0 ± 5.3	12.1 ± 2.5	8.0 ± 1.9 × 164.9 ± 19.0
DIABETES			
MLLKM (GAUSSIAN)	75.3 ± 3.2	0.9 ± 0.5	24.0 ± 5.4 × 1
MLLKM (SQUARE)	74.9 ± 3.6	0.8 ± 0.6	28.1 ± 13.0 × 1
MLLKM (COMPONENT GAUSSIAN)	**76.0** ± 4.1	**2.0** ± 0.9	29.6 ± 7.3 × 1
MLLKM (COMPONENT SQUARE)	72.6 ± 3.5	1.1 ± 1.1	38.6 ± 11.3 × 1
SAG	66.1 ± 9.0	0.0 ± 0.0	1 × 1
LLSVM	**74.2** ± 2.7	**2.3** ± 2.0	32 × 1
LaSVM (GAUSSIAN)	**77.1** ± 3.2	**3.6** ± 1.5	1 × 141.9 ± 7.4
SEQUENTIALMKL	69.5 ± 2.8	55.0 ± 118.6	7.1 ± 9.7 × 216.6 ± 21.7
SIMPLEMKL	69.5 ± 4.1	8.9 ± 1.0	3.3 ± 0.5 × 223.5 ± 4.5

give some guarantees the training procedure always gives a good solution.

To train the MLLKM, we proposed a generic ℓ_1-MKL algorithm that is able to cope with a very high number of kernels. The procedure is based on a reduced active set of kernels and scans the remaining kernels for insertion. As such, 2 extensions can easily be considered: An online kernels MKL where kernels are streamed while the training set is known, and a full online where both training samples and kernels are streamed. This later case is the one of

an online MLLKM since the kernels are build around the training samples.

Finally, our experiments show that MLLKM is able to obtain accuracies comparable to non-linear classifier such as MKL, while having an inference time comparable to linear methods.

References

Amari, Shun-ichi and Wu, Si. Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, 12(6):783–789, 1999.

Bach, Francis R., Lanckriet, Gert R. G., and Jordan, Michael I. Multiple kernel learning, conic duality, and the smo algorithm. In *ICML*, 2004.

Bordes, Antoine, Ertekin, Seyda, Weston, Jason, and Bottou, Léon. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6: 1579–1619, 2005.

Bordes, Antoine, Bottou, Léon, and Gallinari, Patrick. Sgdqn: Careful quasi-newton stochastic gradient descent. *The Journal of Machine Learning Research*, 10:1737–1754, 2009.

Chapelle, Olivier and Rakotomamonjy, Alain. Second order optimization of kernel parameters. *Proc. of the NIPS Workshop on Kernel Learning: Automatic Selection of Optimal Kernels*, 2008.

Dekel, Ofer and Singer, Yoram. Support vector machines on a budget. In *NIPS*, pp. 345–352, 2006.

Ertekin, Seyda, Bottou, Leon, and Giles, C Lee. Nonconvex online support vector machines. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(2): 368–381, 2011.

Fornoni, Marco, Caputo, Barbara, and Orabona, Francesco. Multiclass latent locally linear support vector machines. In *ACML*, pp. 229–244, 2013.

Gönen, Mehmet and Alpaydin, Ethem. Localized multiple kernel learning. In *Proceedings of the 25th international conference on Machine learning*, pp. 352–359. ACM, 2008.

Kloft, Marius, Brefeld, Ulf, Laskov, Pavel, Müller, Klaus-Robert, Zien, Alexander, and Sonnenburg, Sören. Efficient and accurate lp-norm multiple kernel learning. In *Advances in neural information processing systems*, pp. 997–1005, 2009.

Ladicky, Lubor and Torr, Philip. Locally linear support vector machines. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 985–992, 2011.

Picard, David, Thome, Nicolas, and Cord, Matthieu. JKernelMachines: A Simple Framework for Kernel Machines. *Journal of Machine Learning Research*, 14: 1417–1421, May 2013.

Rakotomamonjy, Alain, Bach, Francis, Canu, Stéphane, Grandvalet, Yves, et al. Simplemkl. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

Roux, Nicolas Le, Schmidt, Mark W., and Bach, Francis. A stochastic gradient method with an exponential convergence rate for finite training sets. In *NIPS*, pp. 2672–2680, 2012.

Shalev-Shwartz, Shai and Zhang, Tong. Stochastic dual coordinate ascent methods for regularized loss. *Journal of Machine Learning Research*, 14(1):567–599, 2013.

Shalev-Shwartz, Shai and Zhang, Tong. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In *Proceedings of The 31st International Conference on Machine Learning*, pp. 64–72, 2014.

Shalev-Shwartz, Shai, Singer, Yoram, and Srebro, Nathan. Pegasos: Primal estimated sub-gradient solver for svm. In *ICML*, pp. 807–814, 2007.

Sonnenburg, Sören, Rätsch, Gunnar, Schäfer, Christin, and Schölkopf, Bernhard. Large scale multiple kernel learning. *The Journal of Machine Learning Research*, 7: 1531–1565, 2006.

van Gemert, Jan C, Geusebroek, Jan-Mark, Veenman, Cor J, and Smeulders, Arnold WM. Kernel codebooks for scene categorization. In *Computer Vision–ECCV 2008*, pp. 696–709. Springer, 2008.

Wang, Jinjun, Yang, Jianchao, Yu, Kai, Lv, Fengjun, Huang, Thomas, and Gong, Yihong. Locality-constrained linear coding for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 3360–3367. IEEE, 2010.

Yu, Kai, Zhang, Tong, and Gong, Yihong. Nonlinear learning using local coordinate coding. In *Advances in neural information processing systems*, pp. 2223–2231, 2009.

A. ICML'15 reviews

In this section, we show the reviews received at ICML'15 where the paper was ultimately rejected.

A.1. Reviewer 3

Overall Rating: Strong reject
Reviewer confidence: Reviewer is an expert
Detailed comments for the authors

Summary ——- The paper looks at the problem of accelerated prediction times with non-linear classifiers. To this end the paper proposes a multiple kernel learning (MKL) approach. The base kernels chosen are "locally linear" - to understand these local kernels better, let me take an extreme setting. Each local kernel is centered around an "anchor point" c. For points x, y within a radius r of c, the linear kernel is preserved $k_c(x, y) = (x - c)'(y - c)$. If either point x, y is outside this radius, $k_c(x, y) = 0$.

Of course the paper uses smooth variations using conformal maps to implement these kernels which reweigh the local kernel according to the distance of the points from the anchor point. Radial as well as coordinate-wise maps are considered as well. Every training point is considered a potential anchor point and a sparse combination of these kernels is learnt using the SequentialMKL approach which alternates between updating an active set of kernels and choosing an appropriate combination of these active kernels.

Quality ——- The paper delves upon a slightly stale methodology of "locally linear" methods. These are intuitive methods that are easy to explain but hard to implement and even harder to scale to large problems.

Clarity ——- The paper is well written. Section 4 could do with proof checking to remove spelling/grammatical mistakes.

Originality ——— The idea itself seems novel although it effectively promotes similar classification characteristics as other local methods. The novel part in the paper seems to be its truly non-parametric approach to locality wherein every training point is considered a potential anchor point. Most such works implement locality by clustering the data using k-means and then choosing the cluster centers as anchor points.

Significance ——— The approach seems not very promising practically as recent literature (please see references below) already has several examples of methods that outperform locally linear methods comprehensively in terms of accuracy, as well as prediction time.

Comments ——— - Since a primary goal of the paper seems to be reduced prediction time, recent works in that direction such as (Hsieh et al, 2014) and (Jose et al, 2013) need to be referenced, especially since these methods handily outperform locally linear methods. - The experimental work is a bit underwhelming - all prediction times are in milliseconds. Datasets used are tiny at a time when other such

papers are performing experiments on the MNIST 8Million dataset. Moreover the proposed method does not present a strong case for choosing it over other methods. - Since this is primarily an experimental piece of work with little theoretical results, a much more comprehensive comparison to the state of the art would have to be performed to argue in favor of the proposed method.

References ———

C.-J. Hsieh, S. Si, and I. S. Dhillon, Fast Prediction for Large-Scale Kernel Machines, NIPS 2014.

C. Jose, P. Goyal, P. Aggrwal and M. Varma. Local deep kernel learning for efficient non-linear SVM prediction, ICML 2013.

[REQUIRED AFTER REBUTTAL]
I have read and considered the authors' rebuttal. Yes
Post-rebuttal Comments. I would encourage the authors to support their method with comprehensive experiments on large scale datasets and comparisons to the state-of-the-art.

A.2. Reviewer 4

Overall Rating: Weak reject
Reviewer confidence: Reviewer is knowledgeable
Detailed comments for the authors

In combining multiple locally linear kernels, the idea is to start with one candidate kernel centered on each training point and let regularization pick the ones that are necessary and remove the ones that are not; simiarly for the kernel spread (γ), a bunch of values are made available as candidates for each center and the regularizer keeps the necessary ones. This reads like a nice idea but ideally the position (anchor) can be different from any of the training points (that's the advantage of methods like LLSVM and LMKL) and similarly the ideal spread can be different from any of the predefined candidates. In a high dim problem with a small data set, limiting the anchors to the training data may be restrictive.

Concerning the experimentation: The synthetic problem is simple (and what can you say about the chosen γ?) and comparison is done on four small UCI data sets. The stdev vaues on the latter seem large with respect to differences between the means; are those differences statistically significant? (Why are some results printed in boldface?)

And do not start the abstract with "In this paper"–it is redundant.

[REQUIRED AFTER REBUTTAL]
I have read and considered the authors' rebuttal. Yes
Post-rebuttal Comments. More comparison with existing work is needed. My original review stands.

A.3. Reviewer 6

Overall Rating: Weak reject
Reviewer confidence: Reviewer is an expert
Detailed comments for the authors

The goal of this paper is to improve traditional multiple kernel learning so that to fit the gap of the inference/prediction speed and accuracy between linear SVM and kernel SVM. There are three main contributions: (1) use l1 regularized multi-kernel learning objective aligned with locally linear kernels; (2) speed up the optimization by considering a reduced set of kernels and then probing new kernels after solving MKL in the reduced set; (3) new feature mapping. The experiments are conducted on synthetic and public datasets showing the efficiency of the proposed method. The paper is well written and organized.

I have several questions/suggestions for this paper:

1: It would be great to show the motivation for the new feature mapping in eq(1) or explain the advantage of this feature mapping over traditional kernels(Gaussian kernel or linear kernel). In the experimental part, it would be interesting to vary kernel functions in your MKL framework, e.g., replace feature mapping in eq(1) with linear feature mapping, etc., so that readers can believe that the proposed new feature mapping can benefit the classification accuracy.

2: I guess I am still confused why this method is memory efficient. Would all the kernel matrices be computed and stored in advance to perform kernels selection? I guess the total memory requirement is still $O(n^2)$. Maybe authors can provide more explanation over the memory requirement.

3: There are x_c in the new feature mapping. Do you choose them by kmeans clustering? Or x_c needs to be learned through optimization. Is $c = n$ (the number of data samples)? Will the prediction be faster if using kmeans centers as x_c?

4: Some comparisons with state-of-the-art fast inference algorithms are missing: for example LDKL in [1] and DC-Pred++ in [2]. Also as claimed in the paper, the method is scalable, and it would be interesting to show some experimental results on large-scale datasets. All the datasets tested in the paper have less than 1000 data points.

[1] C. Jose, P. Goyal, P. Aggrwal, and M. Varma, Local deep kernel learning for efficient non-linear svm prediction, in ICML, 2013. [2] C.-J. Hsieh, S. Si, and I.S.Dhillon, Fast Prediction for Large-Scale Kernel Machines, in NIPS, 2014.

5: If I understood correctly, the prediction/inference time for the proposed algorithm is # of kernel * number of support vectors in each kernel. So the fourth column of Table 2 should correspond to the third column of Table 2. For ionosphere dataset, the MLLKM (Gaussian) is with the similar # of kernel * number of support vectors with MLLKM (component gaussian), however the former takes only 1/8 inference time of the latter one. So authors might want to clarify the difference.

[REQUIRED AFTER REBUTTAL]
I have read and considered the authors' rebuttal. No

A.4. Meta-Reviewer 1

Overall Rating: Reject
Detailed Comments The reviewers are in agreement that this paper is not yet ready for publication. All three reviewers consistently felt, both before the reviewer discussion and afterwards, that the paper has major weaknesses that need to be addressed before the paper can be accepted for publication. Both Area Chairs agree with the reviewer assessment. The rebuttal is unconvincing in that DCPred++ should have been cited and discussed in the paper even if experimental comparisons could not have been carried out by the time of submission. Furthermore, some comparative results should have been presented in the rebuttal – it should have been possible to run the proposed algorithm on the DCPred++ data sets even if DCPred++ could not have been implemented by the authors till then. Finally, comparisons should also have been performed to LDKL since both the code and data sets are publically available. The authors should take the reviewer feedback into account while preparing future versions of the manuscript.