Decision Surfaces of Localized Classifiers

CScott Brown and Ryan G Benton
School of Computing
University of South Alabama
Mobile, AL
Email: gitpushoriginmaster@gmail.com

***Abstract*—**
Local learning algorithms are a very general class of non-parametric lazy learners obtained by stitching together systems of locally weighted parametric models. For a system of locally-learned classifiers, there is currently no convenient method to find points on the decision surface. In this paper, we introduce a generic algorithm for finding the decision surface for systems of localized classifiers using arbitrary model families. The decision surface of a classifier is often useful for obtaining pseudo-probabilistic output from the orthogonal distance of a point to the decision surface. We therefore extend our method to find this orthogonal projection of an arbitrary point onto the decision surface for a broad class of classifier families. We specifically derive the necessary equations for computing the orthogonal projection onto the decision surface of systems of locally linear support vector machines. We demonstrate how this can be used for pseudo-probabilistic calibration, and by extension for multiclass classification strategies such as one-vs-rest or one-vs-one. Lastly, we demonstrate the efficacy of this method to obtain more accurate multi-class classifiers on popular datasets.

I. Introduction

Parametric classification methods form a broad class of statistical and machine learning algorithms for data with categorical dependent variables. Typically, parametric methods require stringent assumptions about the distribution of the underlying data, making them inflexible, but providing a limited set of parameters that can be used to interpret the model. On the other hand, nonparametric methods forego these assumptions, making them more flexible, but less interpretable.

Methods for creating a nonparametric model family from a parametric one via model localization are common in the statistics community. A common scheme when the independent variables are continuous is via kernel weighting methods [1], which are a category of "lazy learners" [2]. Nonparametric regression methods evolving from the kernel-weighting paradigm include the Nadaraya-Watson estimator [3], LOESS [4], and local polynomial models [5]. This scheme is not limited to regression, and the same general idea has been extended to clustering problems [6], density estimation [7], anomaly detection [8] and surface reconstruction [9]. More recently, learning local representations of data has become an important part of supervised learning in many domains, including image-processing [10] and time-series analysis [11].

A broad class of local models can be formed by localization of classification models. Many properties of systems of local models employing specifically linear SVMs have been investigated. Work on the topic begins circa 1992 with the seminal work by Bottou and Vapnik [1], who employ linear SVMs with the square KNN kernel. Local modeling remains a popular topic, and improvements to the base algorithm include extensions using the "kernel trick" [12], kernel and bandwidth selection [13], feature weighting [14], efficiency heuristics [15] and others. Interestingly, despite the seemingly large interest in constructing new classifiers by means of systems of local SVMs and other models, to our knowledge no attention has been paid to the discovery of or the properties of the decision surface of the resulting classifier in the input space. This is problematic, since many applications of a classification model family depend on the decision surface. One obvious example is obtaining scores from a classifier, which are often some function of the orthogonal distance of a point to the decision surface. We anticipate that the topic of this investigation will be of use to any future investigations into local SVMs and other varieties of local classifiers.

Although a distance to the *local* decision surface can be readily obtained from the local model at a test point, it is unclear whether these distances can be meaningfully compared between two test points. This problem has repercussions for techniques that rely on score comparisons, such as extending classifiers to multi-class problems using the one-vs-rest or one-vs-one schemes. It is therefore important to be able to find the decision surface of a system of localized classifiers, as distinct from the decision surfaces of the individual local models.

Unfortunately, although the individual local models can be used to obtain predictions from many classification model families, individual models at arbitrary query points q cannot be used to directly compute the decision surface of the system. Recently, algorithms such as Subspace-constrained Mean Shift [9] have been developed that use local models to perform the superficially similar problem of surface reconstruction in point-cloud problems. We will show that this approach can be modified to obtain the decision surface for localized classifiers such as systems of local support vector machines.

The modified Subspace-constrained Mean Shift that we introduce can be used to find points lying on the decision surface, but it is difficult to predict precisely where on the surface the algorithm will terminate. To remedy this, we introduce a simple enhancement to the algorithm that allows resampling of the surface along a particular direction. For well-behaved classifiers, we demonstrate that this resampling technique can be used to find the orthogonal projection of an arbitrary point onto the decision surface.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

Armed with the decision surface, and the orthogonal projections, we can readily apply calibration techniques to obtain pseudo-probabilistic output from the resulting model. We then use the resulting probabilities to tackle multi-class classification problems. To demonstrate the effectiveness of the algorithm, we apply this technique to a simple toy dataset, and also to a handwritten digits classification task [16].

II. Background

A. Local Models

Local regression methods are frequently employed as non-parametric *smoothing* procedures. Such popularly used methods include local constant models (Nadaraya-Watson) [3], local linear models (LOESS) [4], and local polynomials [5]. The procedure used to localize these models can be applied to arbitrary models that allow a weighted learning procedure, and that make predictions [17]. The authors of [1] apply this same concept to obtain predictions from localized linear classifiers.

The general idea of local modeling is to fit a model to a locally-weighted neighborhood of the training data, and then to make a prediction with the obtained local model. If the weighting scheme and loss function of the model family are in C^m, and the loss function is convex, then the parameters of the local models will generally be C^m in the query point as well, resulting in well-behaved predictions across the space. Although this process can apply to many model families, simple models are the most popular candidates for localization, since convexity is a desirable property of the candidate model family, and also since the reduction in training data caused by the localized-weighting step often opposes the construction of complex local models.

Still, other applications for local models than mere predictions have suggested more complex models for localization. In [8], the authors use the model parameters of localized ARMA models for anomaly detection. In [18], the authors use the model parameters of localized Gaussian Process models for time series classification.

We formalize the notion of local models as follows: Suppose a model family F together with some training algorithm A that admits a weighted training scheme. Each training point $x_i \in X \subset D$ is weighted as $w_i \in \mathbb{R}$, so that $A : \mathcal{P}(D \times \mathbb{R}) \to F$ for some domain D. We define the corresponding system of localized models F' to give a single model in F for each point in the domain D, $F' = \prod_{q \in D} F$. The localized algorithm $A'_K : \mathcal{P}(D) \to F'$ extends A by considering the weights $w_i = K(q, x_i)$ to be a function of the data $X \subset D$ and some query point $q \in D$. Thus, A'_K takes in a dataset $X \subset D$, and returns a set of models, one $f_q \in F$ for every point $q \in D$. Some local algorithms, such as LOESS, only require that $q \in D^*$ where D^* is some subspace of independent variables of D, but we will not formalize this distinction in our analysis. In the proceeding, the weight function K and other functions are assumed to handle the division between dependent and independent variables appropriately, and we will abuse the notation accordingly.

If the training algorithm A minimizes a loss function that is the sum of individual model errors:

$$L(X) = \sum_{x \in X} E(x, f) \tag{1}$$

Then the localized algorithm is easy to obtain as:

$$A'_K(X) = \text{argmin}_{f_q \in F} \sum_{x \in X} E(x, f_q) K(q, x) \tag{2}$$

Support vector machines happen to be of this type. $K(q, x)$ is usually called the "kernel", and many choices of kernel are common in the literature. One popular choice is a square KNN-kernel, as employed with locally linear SVMs in [1] and [14].

$$K(x, y) = \begin{cases} 1 & ||x - y||^2 \leq ||x - y_k||^2 \\ 0 & \text{else} \end{cases} \tag{3}$$

where y_k is the kth-nearest training point to x. We will employ the Gaussian Kernel with a fixed radius (the "bandwidth") in our experiments, because it is smooth:

$$K(q, x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{||x-q||^2}{2h^2}} \tag{4}$$

Yet other functions can provide a both smooth and variable-width kernel if desired. Although optimal bandwidth selection is an important problem in local modeling, it is not the focus of this work.

In this notation, the Nadaraya-Watson estimator can be obtained by taking F to be constant functions, and letting E be ordinary least-square errors against the dependent dimensions of x. LOESS can obtained by taking F to be linear functions of the independent dimensions of x, and E as ordinary least-square errors.

Often, a particular use case of a system of localized models has the convenient property that the desired quantity is directly obtainable from the learned model f_q. For example, LOESS is commonly used for scatterplot smoothing, requiring predictions at arbitrary independent variable inputs. Since the f_q returned by training an ordinary least squares model represents a linear function of the independent parts of x, we can easily obtain a prediction from f_q at the query point q by simply evaluating the function.

Unfortunately, it is not always the case that the desired quantity can be directly obtained from the learned model f_q. For example, points on the decision surface of a system of localized classifiers cannot be obtained from the individual f_q at arbitrary points q. This is the problem of the current investigation, the solution to which is inspired by another class of local modeling methods that also suffer from this issue.

B. Mean Shift

Although they incorporate local modeling methods, the Mean Shift [19] and Subspace-constrained Mean Shift [9] algorithms are more typically considered to be a subset of the theory surrounding kernel density estimation [7]. This is to some extent because, although Mean Shift is often used as a clustering algorithm, it is theoretically convenient that it happens also to find the modes of the kernel density function. Likewise, it is theoretically convenient that Subspace-constrained Mean Shift happens to find the "ridges" of the kernel density function [9].

We can also view Mean Shift and variants as algorithms that operate on systems of localized models. The primary difference between these methods and the methods described in the previous section is that the use case does not possess the convenient property that the desired output is directly obtainable from the learned model f_q. MeanShift and variants, rather than seeking properties of arbitrary query points q, seek to find query points that satisfy a particular property. Namely, they seek points q that have 0 error evaluated against their own local regression model. For LOESS, the distinction between dependent and independent variables causes this to be a property at *all* query points. Since the model families of MeanShift variants do not distinguish between independent and dependent parts, these points are slightly more difficult to obtain. For these models, the points that lie on their own regression surface can be found by projecting a point iteratively onto its local regression surface, "chasing down" the regression surface in a sense. It is easy to see that such an iterative method can only terminate at a stationary point of the local modeling/projection algorithm. Whether or not the algorithm ever terminates is another question entirely, and whether or not Mean Shift converges in the general case remains an open question [20].

In the notation introduced above, Mean Shift operates on the family F of constant functions returning a vector of the same dimensionality as D, and with E being total least-square errors against x. We can then obtain a projection from q onto the image of f_q. By iterating this projection scheme, we obtain an algorithm that terminates (if it terminates) in a 0-dimensional set, which is convenient as a clustering algorithm, and which we call Mean Shift. Subspace-constrained Mean Shift can be obtained by taking F to be m-dimensional linear embeddings in D, and letting E be total least-square errors against x. By iteratively projecting a point onto the surface of the local total least-squares regression model, we obtain a point on an m-dimensional surface reconstruction of the input data X [9]. We will replicate this intuition of "chasing down" the local surface and apply it to systems of localized classifiers in the proceeding section.

III. Methodology

In the following, we will refer to properties of an "(individual) local model", in which we mean a single model trained at some query point q. For example, for local classifiers, individual local models each possess a unique decision surface. On the other hand, if we form a new classifier g by combining the predictions of the individual models so that $g(q) = f_q(q)$, then g has a decision surface that is distinct from the decision surface of any individual local model. We distinguish these as properties of the "system of localized models", in which we mean the class of algorithms resulting from the process of training many local models and combining the results in some way.

Algorithm 1 Localized Classifier Decision Surface Projection

```
given $A, K, X, q$
$y \leftarrow q$
**repeat**
    $f_q \leftarrow A_{K(q,X)}$
    $y \leftarrow proj(y, DS(f_q))$
**until** convergence
**return** $y, f_q$
```

For classification models, we can obtain a prediction at a query point q from the individual local models f_q directly, so that nothing further is required whenever predictions are the ultimate goal. It is often desirable to obtain additional properties of the trained model when employing classification models. For example, when extending an algorithm to multiclass problems via a one-vs-rest or one-vs-one strategy, it is necessary to obtain *scores*. For such applications, a natural scoring mechanism is to obtain the orthogonal distances to the decision surface. The decision surface of a system of localized classifiers is not naively given by the decision surfaces of or any other property of the individual f_q at arbitrary query points q.

However, it is easy to see that a point q will lie on the decision surface of the system of localized classifiers if and only if it lies on the local decision surface at q. To be clear, if we let A be some family of simple binary classifiers, then the combined predictor g described above predicts 1, 0 or "decision surface" precisely when f_q does, by definition. If we adopt an iterative projection scheme analogous to Mean Shift variants, then the decision surface of the system is precisely the stationary points of the projection scheme.

Letting $proj(x, Z)$ be the orthogonal projection of $x \in D$ onto $Z \subset D$, and $DS(f)$ the decision surface of f:

Although this algorithm will converge to a point on the decision surface if it converges, the fact that there is currently no general proof for the convergence of Mean Shift [20] causes us to suspect that a proof of convergence of this algorithm is non-trivial. We will therefore forego an attempt at a proof and provide empirical evidence instead, provided in Section IV.

Unfortunately, this scheme generally will not proceed toward the *nearest* point on the decision surface. Therefore, we introduce some additional mechanics to obtain the orthogonal projection. The general idea is to iteratively project a query point toward the decision surface of a single local model defined at that point along some initial vector v. This algorithm terminates (if it terminates) at a point on the global decision surface for the localized classifier in the direction of v if one exists. v is then iteratively adjusted toward the surface normal at that point. The result is the orthogonal projection of the query point onto the decision surface, which can then be used toward, e.g. Platt calibration of the full model.

Suppose that the kernel K is continuously differentiable,

Algorithm 2 Constrained Decision Surface Projection

```
given $A, K, X, q, v$
$y \leftarrow q$
repeat
    $f \leftarrow A_{K(q,X)}$
    $y \leftarrow proj(y, DS(f) \cap \{x | x = y + \lambda v\})$
until convergence
return $y, f$
```

Algorithm 3 Orthogonal Decision Surface Projection

```
given $A, K, X, q, \alpha$
$y, f \leftarrow$ Algorithm $1(A, K, X, q)$
repeat
    $n \leftarrow (\nabla S)(y)$
    $v \leftarrow (1-\alpha)(y-q) + \alpha n$
    $y, f \leftarrow$ Algorithm $2(A, K, X, q, v)$
until convergence
return $y$
```

the error function E is continuously differentiable, and the loss function L is convex. Then the localized classification algorithm A' in Equation 2 is a composition of continuously differentiable functions and is therefore itself continuously differentiable. This is a convenient property, since it implies that the scores obtained from the orthogonal distance from q to $DS(f_q)$ can thus be stitched together into a single continuously differentiable function $f^*(q) = ||q - proj(q, DS(f_q))||$. The implicit function theorem gives that the level set $f^*(q) = 0$ is locally a differentiable function of one of the components of q whenever the gradient is not the zero vector. Note that although some loss functions such as hinge loss are not everywhere differentiable, we only require differentiability along the decision surface. For hinge loss, this will only occur when one of the training data lie exactly on the local decision surface. This is almost guaranteed to occur at some points, but for finite training sets, this set of points will be limited. Thus, even for common non-differentiable loss functions, the normal vectors to the level set exist and are differentiable almost everywhere, allowing us to apply a wide range of optimization strategies to the direction of our projection.

In order to perform various optimization algorithms against the direction of projection, we require a means to resample the decision surface in a orderly fashion. We amend algorithm 1 to allow constraint to a particular direction vector v.

It is important here to choose v so that there exist points on the decision surface in that direction. However, it is easy to check if this algorithm has converged to the decision surface, by simply evaluating $proj(y, DS(f))$ on the result, and ensuring that it gives y.

For classifier families with a linear decision surface, the constrained projection is given by:

$$proj(y, DS(f) \cap \{x | x = y + \lambda v\}) = y - \frac{y \cdot n}{v \cdot n} v$$

where n is the normal vector to the local decision surface. For non-linear classifiers, a means to project onto the decision surface along a particular direction may or may not have such a convenient closed-form.

By varying the vector v along which we make our projection, we can search for a point on the decision surface that is closest to our query point q. If we further require that K and E are twice differentiable, then the normal vectors to the decision surface will also be differentiable, and we can therefore perform ordinary gradient-dependent optimization algorithms on v to minimize its difference with the surface normal. Again, we note that we only require this property along the decision surface, so that loss functions that are twice differentiable almost everywhere are likely to not exhibit problems for non-contrived datasets.

First, we run the above described iterative projection procedure to find *some* point on the decision surface, y_0. Since the level set at that point is a differentiable function within some ϵ ball by the implicit function theorem, we consider the level surface at that point in the reference frame where n_i, the normal vector to the surface at y_i, is the independent variable. The level surface near y_i can thus be written as a function $g(a) \propto n_i$, where a is a vector in the null space of n_i^T. The gradient of the distance between q and y_i can be taken with respect to this new domain to obtain:

$$\frac{\partial}{\partial a} ||q - y_i||^2 = 2rej_{n_i}(y_i - q)$$

where $rej_{n_i}(x)$ is the vector rejection of x onto n_i. Since actually writing our decision surface as a closed-form function is infeasible, we would like to make a small step along the surface in this direction instead. We can accomplish this by constraining the projection step of our iterative algorithm from q to be some convex combination of the vector n_i and $(y_i - q)$. If y_{i+1} is forced to lie along such a line from q, then:

$$\begin{aligned}
&rej_{n_i}(y_{i+1}) = \\
&rej_{n_i}(q + \lambda((1-\alpha)(y_i - q) + \alpha n_i)) = \\
&rej_{n_i}(q) + C_1 * rej_{n_i}(y_i - q) = \\
&rej_{n_i}(y_i) + C_2 * rej_{n_i}(y_i - q) = \\
&rej_{n_i}(y_i) + C_2 * \tfrac{\partial}{\partial a} ||q - y_i||
\end{aligned}$$

where C_j are some constant values. We therefore obtain a simple gradient descent algorithm. This assumes that our constrained iterative projection algorithm actually converges to some point y_i. For an arbitrarily chosen direction, this is not guaranteed. Still, since there exists some ϵ on which our level surface is a function, as long as we constrain α to be small enough that our algorithm does not leave that ϵ ball, the existence of y_i is assured. It is not clear how to find the size of ϵ, so we will satisfy ourselves with simply choosing α to be "small", in the same sense that ordinary gradient descent chooses a "small" step size.

We can now apply the gradient descent process described above to obtain an orthogonal projection onto the decision surface:

where S is the decision surface of our system of local classifiers. Note that if algorithm 3 fails to converge to the decision surface due to having "missed" it via poor selection of v, a simple heuristic to allow the algorithm to continue is to simply apply algorithm 1 to the result. We have found that this generally gives good results.

What remains is a convenient way to compute $(\nabla S)(y)$. One possibility is to perform a finite difference by taking a grid in some small neighborhood of y and applying algorithm 1 to obtain points on the decision surface S. We can then compute a hyperplane estimation from those points with, for example, total least-squares regression, which gives an approximation to $(\nabla S)(y)$.

Finite difference approximations are generally inferior to direct differentiation, which can be computed for most loss functions with a nicely-behaved second derivative. For classifiers with linear decision functions, the derivative has a particularly nice form. Although we limit our proceeding discussion to linear classifiers, much of the reasoning can be applied to classifiers with non-linear decision functions, but must be handled on a case-by-case basis.

Suppose that our classifier family has a linear decision function. Then the distance from q to the projection of q onto the decision surface of the local model centered at q can be written in terms of the parameters of the linear model:

$$f^*(q) = q^T \mathbf{n}^* - \lambda^* \tag{5}$$

where $\mathbf{n}$ and λ are the normal vector to the local decision surface and the offset from the origin, respectively. $\mathbf{n}^*$ and λ^* are the optimal values computed by our training algorithm localized at q. The derivative of this w.r.t q is:

$$\frac{\partial f^*}{\partial q} = \mathbf{n}^* + q^T \frac{\partial \mathbf{n}^*}{\partial q} - \frac{\partial \lambda^*}{\partial q} \tag{6}$$

It is very often the case that $\mathbf{n}^*$ and λ^* are the result of an argmin operation on some loss function L:

$$\mathbf{n}^*, \lambda^* = \text{argmin}_{\mathbf{n},\lambda} L(X, q, \lambda, \mathbf{n})$$

Unfortunately, argmin does not have a convenient derivative. A common trick to differentiate such functions is to rely on the fact that, for differentiable loss functions L, the minimum is obtained only when the derivative w.r.t the argmin variables is 0. If L is differentiable and convex, then:

$$\frac{\partial}{\partial \mathbf{n}} L(X, q, \lambda, \mathbf{n})|_{\mathbf{n}=\mathbf{n}^*} = 0$$

where $\cdot|_{a=b}$ denotes substitution of a with b. Taking a derivative of both sides of this formula w.r.t q yields:

$$\frac{\partial^2 L}{\partial q \partial \mathbf{n}}|_{\mathbf{n}=\mathbf{n}^*} + \frac{\partial^2 L}{\partial \mathbf{n}^2}|_{\mathbf{n}=\mathbf{n}^*} \frac{\partial \mathbf{n}^*}{\partial q} = 0$$

The $\partial^2 L/\partial \mathbf{n}^2$ term is the Hessian of L w.r.t $\mathbf{n}$. If the Hessian is invertible at $\mathbf{n}^*$, we can continue to solve this equation for $\partial \mathbf{n}^*/\partial q$:

$$\frac{\partial \mathbf{n}^*}{\partial q} = -\left(\frac{\partial^2 L}{\partial \mathbf{n}^2}|_{\mathbf{n}=\mathbf{n}^*}\right)^{-1} \frac{\partial^2 L}{\partial q \partial \mathbf{n}}|_{\mathbf{n}=\mathbf{n}^*} \tag{7}$$

A parity of reasoning holds for $\partial \lambda^*/\partial q$:

$$\frac{\partial \lambda^*}{\partial q} = -\left(\frac{\partial^2 L}{\partial \lambda^2}|_{\lambda=\lambda^*}\right)^{-1} \frac{\partial^2 L}{\partial q \partial \lambda}|_{\lambda=\lambda^*} \tag{8}$$

Specifically for SVM classifiers with squared hinge loss, the Hessian above is invertible whenever none of the training data lie on the decision surface of the local model centered at q. We can then compute a nice closed form for ∇S. If we let $SV(X)$ be the set of support vectors for X (i.e. $1 - y(x^T \mathbf{n} - \lambda) > 0$), then the squared hinge loss is given by:

$$L(X, \lambda, \mathbf{n}) = \sum_{x,y \in SV(X)} (1 - y(x^T \mathbf{n} - \lambda))^2 \tag{9}$$

Applying the localization process in equation 2 gives the following loss function for our system of localized SVMs:

$$L(X, q, \lambda, \mathbf{n}) = \sum_{x,y \in SV(X)} (1 - y(x^T \mathbf{n} - \lambda))^2 K(q, x) \tag{10}$$

The derivatives of this w.r.t $\mathbf{n}$, λ and q exist and are straightforward. Employing these and equation 5 directly in equation 7, we obtain:

$$\frac{\partial \mathbf{n}^*}{\partial q} = \left(\sum_{x,y \in SV(X)} x x^T K(q, x)\right)^{-1} * \left(\sum_{x,y \in SV(X)} y x^T \frac{\partial K}{\partial q} - x(x-q)^T \frac{\partial K}{\partial q} \mathbf{n}^{*T}\right)$$

And similarly in equation 8:

$$\frac{\partial \lambda^*}{\partial q} = \left(\sum_{x,y \in SV(X)} K(q, x)\right)^{-1} * \left(\sum_{x,y \in SV(X)} (x-q)^T \frac{\partial K}{\partial q} \mathbf{n}^{*T} - y \frac{\partial K}{\partial q}\right)$$

These can be employed in equation 6 to exactly compute ∇S at q satisfying the above conditions, and lying on the decision surface of the system of localized classifiers. Note that the inverse term in $\partial \mathbf{n}^*/\partial q$ is the inverse of the weighted covariance of the independent parts of X, which is generally expected to be invertible for non-contrived data. Also, if we choose a kernel K that is not everywhere 0, the denominator in $\partial \lambda^*/\partial q$ exists as well.

If the data are well-balanced near q, then these two inverse terms should be comparatively large. On the other hand, the sum-over-y terms should be relatively small since they involve the sum of many comparably-sized positive and negative terms. The sums involving an $(x - q)$ term should also be relatively small, since there will similarly be much canceling out of vectors pointing in various directions around q. Thus, at points that are within the bounding box of the training data, the largest contributor to $\partial f^*/\partial q$ will generally be the unit $\mathbf{n}^*$. We can therefore approximate $(\nabla S)(y)$ with the normal vector of the local decision surface at y. All of our experiments in section IV employ this approximation. Comparing this approximation with the finite difference formula proposed above, we have found that for squared hinge loss SVM, for points on the decision surface within the bounds of the training data, this approximation holds to a high degree of accuracy.

We also yet again note that we only require a non-zero second derivative along the decision surface, so that even loss functions that are not differentiable everywhere may still be differentiable almost everywhere on the decision surface. Our experiments show that the approximation of the normal vector of the decision surface of a system of localized SVMs with hinge loss is reasonably well approximated by the normal vector of the local model at a point lying on the decision surface. We demonstrate this in the proceeding section.

IV. Results

In this section we apply the algorithms described in the previous section with a linear support vector classifier on both a toy dataset and a handwritten digit classification task [16]. We employ the LinearSVC class of the scikit-learn library, which provides a wrapper for the liblinear [21] support vector classifier implementation. Since the loss functions available to SVM classifiers are of the sum-of-individual-errors form, we employ the localization scheme described in equation 2.

A. Toy Data

The toy dataset shown in Figure 1 was generated from the scikit-learn "make_moons" method, using a noise level of 0.06, mean-centered and scaled to have unit variance in both x and y. For our local models, we employ a Gaussian kernel (equation 4) with bandwidth parameter $h = 0.6$. This parameter was cherry-picked to give pleasing results. Note that as $h \to \infty$, the system of local linear SVMs approaches a single global linear SVM. We found that for very low values of h, the resulting vanishing weights caused problems with liblinear, and the individual models would fail. For the α parameter of algorithm 3, we have found that a wide range of values are viable. We have found that α values up to 0.5 result in pleasing convergence properties, and we note that setting α as high as possible will generally enable the algorithm to converge more quickly.

In addition to the training data, Figure 1 shows the decision surface of the system of localized classifiers in red, found via algorithm 3 applied on a grid. The squared hinge loss is employed, resulting in a differentiable decision surface. Also shown are the orthogonal projections of the training data onto the decision surface found via algorithm 3, shown as black vectors. Lastly, the scalar field of orthogonal distances from each point in a dense grid to the red decision surface, as the background color gradient. Since these orthogonal distances

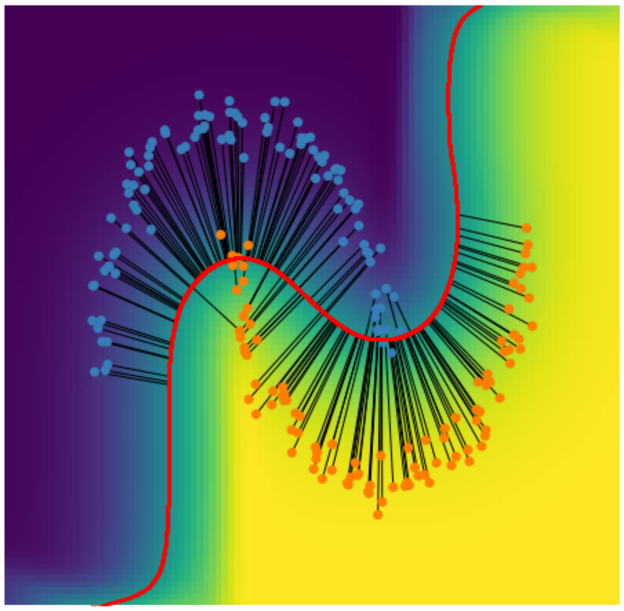

Fig. 1. Local SVM decision surface on toy data, square hinge loss. The scikit-learn "noisy moons" dataset in blue and orange dots. Dot color represents the ground-truth binary label. Decision surface in red. Global scores evaluated on a regular grid in background color gradient. Orthogonal projections of training data onto decision surface in black.

are globally relevant, we can scale these to the unit interval to obtain pseudo-probabilistic output, as we will show in the next section. In these experiments, we have approximated the surface normal at a point q on the surface by the normal of the local SVM centered at q. We also computed a finite difference using a neighborhood of points on the surface for comparison. For query points within the bounding box of the training data, the average cosine similarity between the approximate normal and the finite difference normal was 0.993, with a standard deviation of 0.0079. 99.9% of approximated normals within these bounds had a cosine similarity of at least 0.968. Points outside of the bounds of the training data were less likely to be well approximated. It is well-known that kernel methods tend to have trouble extrapolating, and when moving significantly outside of the range of the training data, the decision surface itself is probably not trustworthy.

We repeat the experiment for the hinge loss, which is not differentiable, to show how this affects the results of the algorithm. Results are in Figure 2. Note that even though the decision surface is not everywhere well-behaved, it is well-behaved enough so that algorithm 3 produces reasonable results for the orthogonal distances. For query points within the bounding box of the training data, the average cosine similarity between the approximate normal and the finite difference normal was 0.911, with a standard deviation of 0.216. 80% of approximated normals within these bounds had a cosine similarity of at least 0.943. Thus, the approximation is not nearly as good for the hinge loss. Nevertheless, algorithm 3 seems to employ it to good effect, as can be

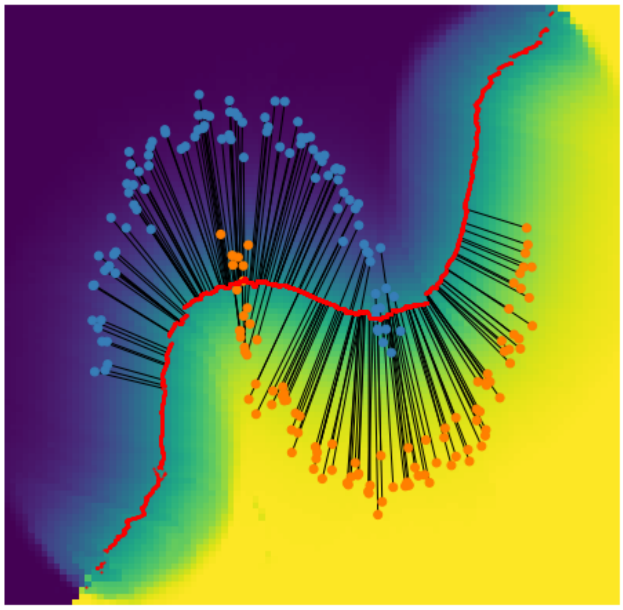

Fig. 2. Local SVM decision surface on toy data, hinge loss. As Figure 1, but employing the hinge loss. The decision surface is not very well-behaved. Still, algorithm 3 gives reasonable estimates for the orthogonal projection.

seen in the figure. As expected, since the hinge loss is not differentiable, the decision surface is not differentiable either. Despite appearances, the decision surface is continuous, but our algorithm for finding orthogonal projections is imperfect for non-differentiable loss functions, and small crags in the surface are therefore difficult to discover. Although the hinge loss provides satisfactory results for individual linear SVMs, because of these problems with systems of localized SVMs utilizing hinge loss, we recommend that researchers and users of localized systems of SVMs consider employing squared hinge loss instead.

B. Handwritten Digits

For this dataset, we employ a one-vs-rest strategy to obtain a multiclass classifier. To this end, we apply Platt Scaling [22] in the ordinary fashion. Due to the small size of the dataset, we have simply set a regularization factor for Platt Scaling a priori at 0.01 rather than using cross-validation to select it automatically. We also set the bandwidth to be fixed a priori at 20 times the average 1-nearest neighbor distance across all training points. We set the SVM regularization to be the library default, which is 1. Choosing better hyperparameters might reasonably improve the results, but we stress that the primary motivation with these experiments is to illustrate a use case of the decision surface for a system of localized classifiers, rather than to earnestly tackle the problem of handwritten digit classification.

We employ here the squared hinge loss to define a system of localized SVMs for each digit label against the rest. For each of these, we compute the orthogonal distance of the training points to the decision surface for the system of localized

TABLE I
LOCAL SVM ONE-VS-REST CLASSIFIER.

	Accuracy	Precision	Recall
Local SVMs	0.961	0.960	0.960
Linear SVM	0.948	0.947	0.948

SVMs. We then train a logistic regression with this orthogonal distance multiplied by the sign of the local prediction at the training point as input. The ground-truth label is taken as output, and the resulting model is used to obtain pseudo-probabilistic scores for each digit. To perform multi-class classification, we compare the probabilities of each model, and choose the digit that scores the highest.

We validate this process on a 30% test split in Table I. Algorithms 1 and 3 were seen to converge for all of our training and test points. We see that this produces modest improvements upon the linear classifier, which we might reasonably expect a non-linear classifier to accomplish. These results demonstrate that computing scores via orthogonal distance to the decision surface of the system of localized classifiers is viable for high-dimensional real world datasets.

V. CONCLUSION

In this paper we describe an algorithm that, if it converges, converges to points lying on the decision surface of a system of localized classifiers. To our knowledge there currently exists no other algorithm to find points on the decision surface. Although we have provided no proof of convergence, and suspect from relationships to other algorithms that such a proof would be difficult to come by, these same relationships and our experiments suggest that the algorithm has nice convergence properties nonetheless.

We further extend this algorithm to find points on the decision surface along an arbitrary direction. This improves upon the naive algorithm which only finds an arbitrary point on the decision surface. We have demonstrated that this algorithm can be used to resample points on the decision surface in an orderly fashion.

We use this algorithm to develop a gradient-descent technique for finding the orthogonal projection of a point onto the decision surface. We demonstrate how the orthogonal projection, and the corresponding orthogonal distance, can be used as scores for pseudo-probabilistic regularization on both toy data and real-world datasets.

For the future, we conjecture that the intuition behind both algorithm 2 and algorithm 3 would work equally well as an extension to Subspace-constrained Mean Shift. These could be used to resample grids over principal surfaces so that a triangular mesh might be obtained, or to find orthogonal projections onto the reconstructed surface which "is one of the most critical operations in computer aided geometric design and applications" [23]. Other future work would involve techniques to improve the efficiency of the proposed algorithms. For example, memoization might be employed to more quickly find nearly-orthogonal points on the decision surface. Also, we suspect that adapting the resampling algorithm to start at points closer to the decision surface would provide significant speed improvement.

References

[1] Léon Bottou and Vladimir Vapnik. "Local learning algorithms". In: *Neural Computation* 4.6 (1992), pp. 888–900.

[2] Christopher G Atkeson, Andrew W Moore, and Stefan Schaal. "Locally weighted learning". In: *Lazy Learning*. Springer, 1997, pp. 11–73.

[3] Elizbar A Nadaraya. "On estimating regression". In: *Theory of Probability & Its Applications* 9.1 (1964), pp. 141–142.

[4] William S Cleveland. "Robust locally weighted regression and smoothing scatterplots". In: *Journal of the American Statistical Association* 74.368 (1979), pp. 829–836.

[5] J. Fan and I. Gijbels. *Local Polynomial Modelling and Its Applications: Monographs on Statistics and Applied Probability 66*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1996. ISBN: 9780412983214. URL: https://books.google.com/books?id=BM1ckQKCXP8C.

[6] Keinosuke Fukunaga and Larry Hostetler. "The estimation of the gradient of a density function, with applications in pattern recognition". In: *IEEE Transactions on Information Theory* 21.1 (1975), pp. 32–40.

[7] George R Terrell, David W Scott, et al. "Variable kernel density estimation". In: *The Annals of Statistics* 20.3 (1992), pp. 1236–1265.

[8] Stephen Bay et al. "A framework for discovering anomalous regimes in multivariate time-series data with local models". In: *Symposium on Machine Learning for Anomaly Detection*. 2004.

[9] Umut Ozertem and Deniz Erdogmus. "Locally defined principal curves and surfaces". In: *Journal of Machine Learning Research* 12.Apr (2011), pp. 1249–1286.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in Neural Information Processing Systems*. 2012, pp. 1097–1105.

[11] Lexiang Ye and Eamonn Keogh. "Time series shapelets: a new primitive for data mining". In: *Proceedings of the 15th ACM International Conference on Knowledge Discovery and Data Mining*. ACM. 2009, pp. 947–956.

[12] Hao Zhang et al. "SVM-KNN: Discriminative nearest neighbor classification for visual category recognition". In: *Proceedings of the 2006 IEEE Conference on Computer Vision and Pattern Recognition*. Vol. 2. IEEE. 2006, pp. 2126–2136.

[13] Liva Ralaivola and Florence d'Alché-Buc. "Incremental support vector machine learning: A local approach". In: *International Conference on Artificial Neural Networks*. Springer. 2001, pp. 322–330.

[14] Vojislav Kecman and J Paul Brooks. "Locally linear support vector machines and other local models". In: *Proceedings of the 2010 International Joint Conference on Neural Networks*. IEEE. 2010, pp. 1–6.

[15] Thanh-Nghi Do and François Poulet. "Random local SVMs for classifying large datasets". In: *International Conference on Future Data and Security Engineering*. Springer. 2015, pp. 3–15.

[16] Ethem Alpaydin and Cenk Kaynak. "Cascading classifiers". In: *Kybernetika* 34.4 (1998), pp. 369–374.

[17] Vladimir Vapnik. "Principles of risk minimization for learning theory". In: *Advances in Neural Information Processing Systems*. 1992, pp. 831–838.

[18] CScott Brown and Ryan Benton. "Local Gaussian Process Features for Clinical Sensor Time Series". In: *Proceedings of the 2019 IEEE International Conference on Bioinformatics and Biomedicine*. IEEE. 2019, pp. 1288–1295.

[19] Dorin Comaniciu and Peter Meer. "Mean shift: A robust approach toward feature space analysis". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 5 (2002), pp. 603–619.

[20] Youness Aliyari Ghassabeh, Tamás Linder, and Glen Takahara. "On some convergence properties of the subspace constrained mean shift". In: *Pattern Recognition* 46.11 (2013), pp. 3140–3147.

[21] Rong-En Fan et al. "LIBLINEAR: A library for large linear classification". In: *Journal of Machine Learning Research* 9.Aug (2008), pp. 1871–1874.

[22] John Platt. "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods". In: *Advances in Large Margin Classifiers* 10.3 (1999), pp. 61–74.

[23] Kwanghee Ko and Takis Sakkalis. "Orthogonal projection of points in CAD/CAM applications: an overview". In: *Journal of Computational Design and Engineering* 1.2 (2014), pp. 116–127.