Neural Decoding with Kernel-based Metric Learning

Austin J. Brockmeier[1], **John S. Choi**[2], **Evan G. Kriminger**[1], **Joseph T. Francis**[2, 3], **and Jose C. Principe**[1]

[1]Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, U.S.A.

[2]Joint Program in Biomedical Engineering, NYU Polytechnic School of Engineering and SUNY Downstate, Brooklyn, NY 11203, U.S.A.

[3]Department of Physiology and Pharmacology, State University of New York Downstate Medical Center, Robert F. Furchgott Center for Neural & Behavioral Science, Brooklyn, NY 11203, U.S.A.

Keywords: Decoding, dependence, kernels, metric learning, spike trains

Abstract

When studying the nervous system, the choice of metric for the neural responses is a pivotal assumption. For instance, a well-suited distance metric enables us to gauge the similarity of neural responses to various stimuli and assess the variability of responses to a repeated stimulus—exploratory steps in understanding how the stimuli are encoded neurally. Here we introduce an approach where the metric is tuned for a particular neural decoding task. In particular, neural spike train metrics have been used to quantify the information content carried by the timing of action potentials. While a number of metrics for individual neurons exist, a method to optimally combine single-neuron metrics into multi-neuron, or population-based, metrics is lacking. We pose the problem of optimizing multi-neuron metrics and other metrics using centered alignment, a kernel-based dependence measure. The approach is demonstrated on invasively recorded neural data consisting of both spike trains and local field potentials. The experimental paradigm consists of decoding the location of tactile stimulation on the forepaws of anesthetized rats. We show that the optimized metrics highlight the distinguishing dimensions of the neural response, significantly increase the decoding accuracy, and improve non-linear dimensionality reduction methods for exploratory neural analysis.

1 Introduction

Systematic analysis of neural activity is used to investigate the brain's representation of different conditions such as stimuli, actions, intentions, cognitive states, or affective states. The fundamental questions of the analysis are "How much information does the neural response carry about the condition?" and "How is this information represented in the neural response?" Information theory can be used to assess the first question, but in practice its application is non-trivial without answering the second question first, due to the challenge in estimating information-theoretic quantities for neural data.

As neural signals vary across time and space, many electrodes and high-sampling rates are necessary to accurately record the spatiotemporal activity. The neural responses may be recorded in different forms either invasively or non-invasively. Modern electrode arrays allow invasive recordings to capture both the timing of action potentials (spike trains) across many neurons, and local field potentials (LFPs) across many electrodes.

Estimating decoding models, let alone an information-theoretic quantity such as mutual information, is difficult on this diverse, high-dimensional data. Consequently, it is necessary to explicitly optimize an alternative, possibly low-dimensional, representation of the neural signals where it is easier to gauge their information content relevant to the experimental condition. At this point it is necessary to make the distinction between the objective here—*learning the representation of the neural response that is most relevant for the different conditions*—and the neural coding problem—*learning the representation of the stimulus space that is most relevant for the neural response*.

An optimized representation of the neural response is useful for visualizing the single-trial variability (Churchland et al., 2010) and the similarity between neural responses to similar conditions (Brockmeier et al., 2013). The neural response representation may be optimized in either a supervised or unsupervised manner.

A number of unsupervised methods have been used for the exploratory analysis of neural data (Stopfer et al., 2003; Broome et al., 2006; Brockmeier et al., 2010, 2011; Park et al., 2012). Principal component analysis (PCA) and clustering are among the simplest, but the results from PCA may be unsatisfactory on neural data (Cowley et al., 2012). Other non-linear approaches include distance embeddings (Sammon, 1969), kernel-based extension to PCA (Schölkopf et al., 1998), and manifold learning algorithms that try to preserve the similarity structure between samples in a low-dimensional representation. Methods tend to concentrate on preserving either local (Roweis and Saul, 2000) or structural-based (Tenenbaum et al., 2000) similarities; novel samples can be mapped to the representation space via explicit mappings (Bunte et al., 2012).

State-space models provide an unsupervised approach to explore the temporal evolution and variability of neural responses during single trials, and those with low-dimensional or discrete state variables (hidden Markov models) are useful to visualize the dynamics of the neural response (Radons et al., 1994; Seidemann et al., 1996; Kemere et al., 2008; Yu et al., 2009a,b). Petreska et al. (2011) have shown how a combination of both temporal dynamics and a discrete state can efficiently capture the dynamics in a neural response. Henceforth, we only consider static representations of responses to a discrete number of conditions, or classes, collected over multiple trials.

In the supervised case, Fisher discriminant analysis (FDA) and extensions (Fuku-

naga, 1990; Baudat and Anouar, 2000; Sugiyama, 2007) use sample covariances from each class to form discriminative projections. The optimal projection is a solution to a generalized eigenvalue problem that maximizes the spread between samples in different classes while minimizing the spread of samples within the same class.

The dimensionality of the neural response can be reduced by feature selection (Kira and Rendell, 1992). The simplest approach is to find how informative each feature is for a given task and then select a set of informative, but not redundant features (Guyon and Elisseeff, 2003; Peng et al., 2005; Yamada et al., 2013), or a set of features may be obtained by backward or forward-selection algorithms (Song et al., 2007, 2012).

In a broad sense, linear projections and dimensionality reduction are just special cases of metric-learning algorithms, which change the metric in the original space to achieve some specific objective, usually exploiting supervisory information (Lowe, 1995; Xing et al., 2003; Fukumizu et al., 2004). Often, metric learning is used as an intelligent preprocessing for classification methods that depend on a measure of similarity or dissimilarity to determine if two samples are of the same class. For instance, kernel machines or nearest-neighbor-based approaches compare novel samples relative to already observed samples but rely on a predefined similarity measure. These classifiers are highly dependent on the preprocessing and offer little insight into the importance of individual feature dimensions. Metric learning can improve the classification performance by adjusting the importance of individual features for the task (Lowe, 1995; Takeuchi and Sugiyama, 2011), and these weights can be used to highlight the features or dimensions relevant for the objective. Furthermore, metric learning approaches can also improve kernel regression (Fukumizu et al., 2004; Navot et al., 2006; Weinberger and Tesauro, 2007).

We investigate classes of metrics that are parametrized along spatiotemporal dimensions of neural responses. For instance, it is natural to consider which channels or time points in the neural response are most useful for distinguishing among conditions. Unlike previous metric-learning approaches that have concentrated on learning projections and weightings for scalar-valued variables, here we also explore using metric learning where the weights correspond to different neurons in multiunit spike train metrics or vectors of spatial amplitudes from multichannel LFPs.

With vector-valued data each individual metric is defined over a vector space. Using a weighted combination of these metrics we can form strictly spatial or temporal weightings for multivariate time series. In addition, we propose to optimize multi-neuron spike train metrics (Aronov, 2003; Houghton and Sen, 2008) formed as combinations of spike train metrics defined for individual neurons (Victor and Purpura, 1996; Van Rossum, 2001; Paiva et al., 2009). To our knowledge, ours is the first attempt to explicitly optimize the parameters of multi-neuron spike train metrics, instead of using pre-defined weightings.

Given the form of the projections or weightings, one must consider their optimization. A number of metric-learning cost functions have been posed in the literature, but we propose using a kernel-based measure of dependence known as centered alignment (Cortes et al., 2012). Centered alignment was shown to be a useful measure for kernel-learning (Cortes et al., 2012), and a similar but unnormalized kernel dependence measure, Hilbert Schmidt information criterion (HSIC), has been used for feature selection (Song et al., 2012). Another kernel-based dependence measure formulated based

on conditional entropy (Sanchez Giraldo and Principe, 2013) has also been shown to be useful for learning a Mahalanobis distance (Sanchez Giraldo and Principe, 2013; Brockmeier et al., 2013). The objective and optimization techniques used here are most similar to those proposed by Fukumizu et al. (2004), but by replacing the kernel-based canonical correlation measure (Bach and Jordan, 2003) with centered kernel alignment we avoid both matrix inversion and regularization.

Using the kernel-based objective, we highlight the connection between optimizing weighted tensor product kernels and metric learning. Optimizing a metric in a kernel-based framework has the added benefit that it naturally optimizes the kernel itself for use in support vector machines. This eliminates the need for the user to choose a kernel size through cross-validation or trial-and-error. Kernels also provide a straightforward way to form metrics corresponding to nonlinear projections. This is done by retrieving a metric from a unweighted sum of optimized kernels—an approach distinct from optimizing a convex sum of kernels (Lanckriet et al., 2004). Ultimately, this leads us to propose optimized multi-neuron spike train kernels formed as the product and the sum of product of single-unit spike train kernels (Paiva et al., 2009; Park et al., 2012, 2013).

The rest of the paper is organized as follows: we first introduce the mathematical representation of the neural response and different metrics that use linear projections or weightings; from the metrics we form kernel-based similarity measures; from the kernels we introduce the dependence measure (Cortes et al., 2012); and from these we form metric-learning optimization problems. We verify the classification performance of the proposed approach on benchmark datasets, and show results on experimental datasets consisting of both local field potentials (LFPs) and spike trains. We conclude with a discussion of the results, the connection of metric-learning to neural encoding, and future applications.

2 Metrics and Similarity Functions

2.1 Neural Data Representation and Metrics

For the trial-wise classification of different conditions, a sample from each trial is the concatenation of the neural response across all selected time samples, electrode channels, or neural spiking units. Let $x = [x_{(1)} \dots x_{(P)}]$ denote the P-dimensional neural response to a given trial, where parenthetical subscripts denote the response dimension: $x_{(i)}$ may be a scalar, vector, or set (in the case of spike trains). Let x_j denote the neural response for the jth trial, $j \in \{1, \dots, n\}$, and let $l_j \in \{1, \dots, m\}$ denote the discrete class label corresponding to a certain class condition for the jth trial. The set $\{z_j = (x_j, l_j)\}_{j=1}^{n}$ represents a joint sample of the neural responses and labels.

Consider the distances between pairs of neural responses along each dimension: the distance between samples x, x' on the ith dimension is denoted $d_i(x_{(i)}, x'_{(i)})$. For instance, this may be the Euclidean metric for scalars or vectors or a metric on spike trains (Van Rossum, 2001; Paiva et al., 2009; Dubbs et al., 2010).

A metric for the joint neural response is formed by combining these individual distances using a feature weighting (Lowe, 1995; Takeuchi and Sugiyama, 2011), where the weights control the importance of the distance along each dimension. Let w denote

a nonnegative P-dimensional weighting vector, such that $\forall i, w_i \geq 0$. The metric using this weighting is formed as

$$d_w^\gamma(x, x') = \sum_{i=1}^{P} w_i d_i^\gamma(x_{(i)}, x'_{(i)}), \tag{1}$$

where the exponent $\gamma \geq 1$ controls how relatively large distances on individual dimensions contribute to the total distance. This is a weighted Euclidean metric if $\gamma = 2$ and the metric for each dimension is the Euclidean or $\mathcal{L}_2$ metric.

If $w_i = 0$ then the metric is actually a pseudo-metric since it does not satisfy the property that $d(x, x') = 0$ if and only if $x = x'$. However, this invariance to certain dimensions is a goal of metric learning, and we will no longer make the distinction between pseudo-metrics and metrics. For vector-valued data the weighting is a special case of a linear transformation that is equivalent to globally scaling the input dimensions: $w_i d_i^\gamma(x_{(i)}, x'_{(i)}) = d_i^\gamma(w_i^{1/\gamma} x_{(i)}, w_i^{1/\gamma} x'_{(i)})$.

If each neural response is a vector of scalars, we can define a more general Euclidean metric parametrized by a matrix $A \in \mathbb{R}^{P \times Q}$ using a linear projection of the samples $y = A^{\mathrm{T}} x, \quad y_{(j)} = \sum_i A_{i,j} x_{(i)}, \ \ j = 1, \ldots, Q$. The Euclidean distance in the Q-dimensional feature space

$d^2(y, y') = \sum_{j=1}^{Q} \|y_{(j)} - y'_{(j)}\|_2^2$

is equivalent to a Mahalanobis distance in the original space, with the inverse covariance matrix replaced by the symmetric positive definite matrix AA^{T}:

$$d_A^2(x, x') = d^2\left(A^{\mathrm{T}} x, A^{\mathrm{T}} x'\right) = \left\|A^{\mathrm{T}} x - A^{\mathrm{T}} x'\right\|_2^2 = (x - x')^{\mathrm{T}} A A^{\mathrm{T}} (x - x'). \tag{2}$$

As A has $P \cdot Q$ coefficients there are more degrees of freedom to distort the space according to this metric. This matrix is *strictly* positive definite if $P = Q$ and AA^{T} is full rank; otherwise, the metric is actually a pseudo-metric.

The special case of a weighted metric (1) appears if A is diagonal and square, with P diagonal entries $A_{i,i} = \sqrt{w_i}$. More generally, a Mahalanobis distance can be seen as a weighting over a squared distance matrix between all dimensions. Using properties of the trace

$$d_A^2(x, x') = \mathrm{tr}\left[AA^{\mathrm{T}}(x - x')(x - x')^{\mathrm{T}}\right] = \mathrm{tr}\left[AA^{\mathrm{T}} D\right] = \sum_{i,j} [AA^{\mathrm{T}}]_{i,j} D_{i,j} \tag{3}$$

where $D_{i,j} = \|x_{(i)} - x_{(j)}\|_2^2 = \langle x_{(i)}, x_{(j)} \rangle - \langle x_{(i)}, x'_{(j)} \rangle - \langle x'_{(i)}, x_{(j)} \rangle + \langle x'_{(i)}, x'_{(j)} \rangle$. Unless A is diagonal this metric exploits the inner-product *between different* dimensions. Written in this form, it is clear how a Mahalanobis-type metric can be formed whenever all the dimensions of the neural response correspond, or can be mapped, to the *same* Hilbert space. Specifically, let $\phi : \mathcal{X} \to \mathcal{H}$ define a mapping from any element $x_{(i)} \in \mathcal{X}$ to an element in the Hilbert space $\phi(x_{(i)}) \in \mathcal{H}$. The Mahalanobis-type metric in this space is defined by (3) where $D_{i,j} = \langle \phi(x_{(i)}), \phi(x_{(j)}) \rangle - \langle \phi(x_{(i)}), \phi(x'_{(j)}) \rangle - \langle \phi(x'_{(i)}), \phi(x_{(j)}) \rangle + \langle \phi(x'_{(i)}), \phi(x'_{(j)}) \rangle$. As long as the inner-product can be defined between the dimensions, for instance by using the spike-train kernels discussed in Section 2.3, one can form metrics that use the distance between different spiking units. This would replicate the interaction between spikes on different units intrinsic to some multi-unit metrics

(Aronov, 2003). However, evaluating the inner-product between each pair of dimensions for every pair of samples is computationally demanding, and is not investigated here.

2.2 Kernels for neural responses

Kernel functions are bivariate measures of similarity based on the inner-product between samples embedded in a Hilbert space. Let the domain of the neural response be denoted $\mathcal{X}$ and consider a kernel $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. If κ is positive definite then there is an implicit mapping $\phi : \mathcal{X} \to \mathcal{H}$ that maps any element $x \in \mathcal{X}$ to an element in the Hilbert space $\phi(x) \in \mathcal{H}$ such that $\kappa(x, x') = \langle \phi(x), \phi(x') \rangle$.

As we want to explore the similarity across the individual dimensions of the data, we compose a joint similarity measure from the marginal similarity on each dimension. Let $\mathcal{X}_i$ denote the neural response domain of the ith dimension and consider a positive-definite kernel $\kappa_i : \mathcal{X}_i \times \mathcal{X}_i \to \mathbb{R}$ and corresponding mapping $\phi_i : \mathcal{X}_i \to \mathcal{H}_i$ for this dimension. The similarity between a pair of samples x and x' on the ith dimension is $\kappa_i(x_{(i)}, x'_{(i)}) = \langle \phi_i(x_{(i)}), \phi_i(x'_{(i)}) \rangle$.

The joint similarity over both dimensions i and j is computed by taking the product between the kernel evaluations $\kappa_{[ij]}(x_{(i,j)}, x'_{(i,j)}) = \kappa_i(x_{(i)}, x'_{(i)}) \cdot \kappa_j(x_{(j)}, x'_{(j)})$. The new kernel $\kappa_{[ij]}$ is called a tensor-product kernel since it corresponds to using a mapping function that is the tensor-product between the individual mapping functions $\phi_{[ij]} = \phi_i \otimes \phi_j$ where $\phi_{[ij]}(x_{(i,j)}) \in \mathcal{H}_{[ij]}$. The product of positive-definite kernels is positive definite, and taking the product over all dimensions returns a positive-definite kernel over the joint space: $\kappa(x, x') = \prod_i \kappa_i(x_{(i)}, x'_{(i)})$.

Due to the product, if for one dimension $\kappa_i(x_{(i)}, x'_{(i)}) \approx 0$ then $\kappa(x, x') \approx 0$. If some of the dimensions are noisy with respect to the task, then they will have a deleterious effect on the joint similarity measure. In order to separately weight the contribution of each dimension in the product, consider taking the kernel for the ith dimension to the $\theta_i \geq 0$ power $[\kappa_i(x_{(i)}, x'_{(i)})]^{\theta_i}$. As $\theta_i \to 0$ the influence of ith dimension decreases, and $\theta_i = 0 \implies (\kappa_i(x_{(i)}, x'_{(i)}))^{\theta_i} = 1$, thereby removing its effect altogether. Taking the product over all dimensions results in a weighted product kernel over the joint space:

$$\kappa_\theta(x, x') = \prod_i \left[\kappa_i(x_{(i)}, x'_{(i)})\right]^{\theta_i}, \tag{4}$$

where $\theta = [\theta_i \dots \theta_P]$ denotes the nonnegative parameter vector. However, not all positive-definite kernels can be taken to an arbitrary power and still be positive definite. Only the class of positive-definite kernels that are infinitely divisible (Horn, 1967) can be taken to arbitrary powers such that the resulting kernel κ_θ is positive definite.

There are many infinitely divisible kernels, but our interest in metric learning leads us to the special case of kernels that are functions of distances $\kappa(x, x') = f(d(x, x')) = f(u)$. Here we rely on the work of Schoenberg (1938) who explored the connection between distance metrics and positive-definite kernel functions: a kernel that is a function of distance metric is only positive definite if the metric space can be isometrically embedded in Hilbert space. From Schoenberg (1938) Theorem 4, the most general function $f(u)$ which is bound away from zero and whose positive powers $[f(u)]^\lambda, \lambda > 0$ are

positive definite is of the form $f(u) = \exp(c + \psi(u))$ where $\psi(u)$ is positive definite and c is a constant. For kernels of this form, positive powers simply scale the constant and function $[f(u)]^\lambda = [\exp\{c + \psi(u)\}]^\lambda = \exp\{c' + \lambda\psi(u)\}, \lambda > 0$.

Thus, a class of kernels whose positive powers are all positive definite are of the form $\kappa(x, x') = f(d(x, x')) = \exp(c + h(x, x'))$ where $h(x, x')$ is positive definite. Given a metric $d(x, x')$, $\kappa(x, x') = \exp(0 + h(x, x')) = \exp\{-g(d(x, x')\}$ is positive definite for only certain choices of $g(\cdot)$. In particular if $d_p(x, x')$ corresponds to a p-norm or an $\mathcal{L}_p$ metric then $\kappa(x, x') = \exp(-d_p^p(x, x'))$ is positive definite for $0 < p \leq 2$ (Schoenberg, 1938). Furthermore, the kernel $\kappa(x, x') = \exp(-d_p^\gamma(x, x'))$ is positive definite for $0 < p \leq 2$ and $0 < \gamma \leq p$ (Schoenberg, 1938). For $p < 1$, d_p is not actually a metric since it violates the triangle inequality; nonetheless, d_p^γ is embeddable in a vector space for $0 < \gamma \leq p$.

Clearly, the Gaussian kernel $\kappa(x, x') = \exp(-\theta d^2(x, x'))$ is positive definite if and only if $d(x, x')$ is a Euclidean or $\mathcal{L}_2$ metric; whereas, the Laplacian kernel $\kappa(x, x') = \exp(-\theta d(x, x'))$ is positive definite for an $\mathcal{L}_p$ metric with $1 \leq p \leq 2$. For kernels of this form, $\kappa_i^{\theta_i}(x_{(i)}, x'_{(i)}) = \exp(-\theta_i d^\gamma(x_{(i)}, x'_{(i)}))$, and substituting this equation into the weighted product kernel (4) yields

$$\kappa_\theta(x, x') = \prod_{i=1}^{P} \exp\left(-\theta_i d^\gamma(x_{(i)}, x'_{(i)})\right) = \exp\left(-\sum_{i=1}^{P} \theta_i d^\gamma(x_{(i)}, x'_{(i)})\right), \tag{5}$$

where θ_i can now be regarded as a parameter of kernel κ_i. Letting $\theta = w$ we have $\kappa_\theta(x, x') = \exp\left(-d_w^\gamma(x, x')\right)$; this shows the equivalence between the weighted metric (1) and parametrized product kernel (4).

Similarly, using the Mahalanobis metric (2) on scalar-valued data, a multivariate Gaussian kernel can be defined as the product of Q Gaussian kernels:

$$\kappa_A(x, x') = \exp\left(-d_A^2(x, x')\right) = \prod_{j=1}^{Q} \exp\left(-d^2\left(y_{(j)}, y'_{(j)}\right)\right), \tag{6}$$

where $y_{(j)} = \sum_i A_{i,j} x_{(i)}$.

For scalar-valued data the weighted and Mahalanobis metrics correspond to linear projections. A nonlinear metric can be formed from the direct sum of kernels—as the sum of positive-definite functions is itself a positive-definite function. Let $\Theta = [\theta^1, \theta^2, \ldots, \theta^Q]$ denote a matrix of different weighting vectors corresponding to a set of product kernels $\{\kappa_{\theta^j}\}_{j=1}^Q$. Define κ_Θ as an unweighted sum of Q product kernels:

$$\kappa_\Theta(x, x') = \sum_{j=1}^{Q} \kappa_{\theta^j}(x, x') = \sum_{j=1}^{Q} \prod_{i=1}^{P} \exp\left(-\theta_i^j d^\gamma(x_{(i)}, x'_{(i)})\right). \tag{7}$$

Let ϕ_Θ denote the implicit mapping defined by the sum kernel. This mapping defines a metric between x and x' that corresponds to the $\mathcal{L}_2$ distance in the Hilbert space

$$d_\Theta(x, x') = \|\phi_\Theta(x) - \phi_\Theta(x')\|_2 = \sqrt{\kappa_\Theta(x, x) - 2\kappa_\Theta(x, x') + \kappa_\Theta(x', x')}. \tag{8}$$

In terms of a group of samples, the γ power of the distance matrix for the ith dimension is denoted $D_i^{\circ\gamma}$ where $[D_i^{\circ\gamma}]_{j,k} = d^\gamma(x_j(i), x_k(i)) \;\; j, k \in \{1, \ldots, n\}$. The notation

$D^{\circ 2}$ denotes that each element is squared $D^{\circ 2} = D \circ D$ where $\circ$ denotes the entry-wise (Hadamard) product, as opposed to the matrix product $D^2 = DD$.

The kernel matrix for the ith dimension is $K_i = \exp(-\theta_i D_i^{\circ\gamma})$. The kernel matrix for the product and sum kernels are computed as $K_\theta = K_1 \circ K_2 \circ \cdots \circ K_P = e^{-\sum_i \theta_i D_i^{\circ\gamma}}$ and $K_\Theta = K_{\theta^1} + K_{\theta^2} + \cdots K_{\theta^Q}$. The labels of the trials can also be represented by a kernel matrix L, where each entry $L_{j,k} = \delta(l_j, l_k)$ use the 0-1 kernel, $\delta(l, l') = 1$ if $l = l'$ and $\delta(l, l') = 0$ if $l \neq l'$.

2.3 Neural Metrics

Out of the many possible neural response metrics, we consider the following metrics:

1) Temporal metrics for multivariate time-series: Each individual distance is the Euclidean distance between the vectors of instantaneous amplitudes across the channels. Each weight corresponds to a particular time lag. The weight adjusts the importance of the distance between the spatial patterns of the two samples at that particular time.

2) Spatiotemporal projections: A linear projection matrix is used to form a Mahalanobis distance.

3) Spike train metrics: Each individual distance is between spike trains on the same unit at different temporal precisions. The weight adjusts the importance of each unit at a particular temporal precision value. There are a number of spike train metrics, but we consider two different metrics:

   A) Spike train alignment metric. The metric is the $\mathcal{L}_1$ or $\mathcal{L}_2$ version of the Victor-Purpura (VP) spike train distance (Dubbs et al., 2010; Victor and Purpura, 1996).

   B) Kernel-based spike metric. The metric is defined by the mapping ϕ induced by a spike train kernel (Park et al., 2012, 2013). We use the memoryless cross-intensity (mCI) spike train kernel (Paiva et al., 2009). Let $x = \mathcal{T}$ and $x = \mathcal{T}'$ be two sets of spike times, the kernel is defined as

$$\kappa(x, x') = \langle \phi(x), \phi(x') \rangle = \sum_{t \in \mathcal{T}} \sum_{t' \in \mathcal{T}'} \exp(-q|t - t'|).$$

   Then $\|\phi(x) - \phi(x')\|_2 = \sqrt{\kappa(x, x) - 2\kappa(x, x') + \kappa(x', x')}$ is an $\mathcal{L}_2$ metric (Paiva et al., 2009).

Alternatively, multichannel spike trains can be transformed to vectors in Euclidean space. First the spike timings for each unit, are quantized into fixed-width, contiguous, and non-overlapping bins. Then the binned spike count vectors for each neuron are concatenated and a spatiotemporal projection can be applied.

Based on these metrics we use kernel functions as measures of similarity. On each individual dimensions we use either the Gaussian kernel for the Euclidean and $\mathcal{L}_2$ distances or the Laplacian for $\mathcal{L}_1$ metrics such as the original Victor-Purpura metric. The

kernels for individual dimensions are combined using the tensor product kernel (5). The sum of product kernels (7) consists of an unweighted sum of the weighted product kernels with different weightings. For the Mahalanobis metric (2) a multivariate Gaussian kernel is used (6).

3 Kernel-based Metric Learing

We introduce a kernel-based measure to quantify the joint information between neural responses and labels corresponding to stimuli or condition. The measures can be used as an objective function to optimize the metric used to evaluate the similarity among neural responses.

3.1 Kernel-based Measures of Dependence

Kernel target alignment measures the similarity between two kernel functions using their normalized inner-product (Cristianini et al., 2002). For jointly sampled data, the inner-product of kernel functions defines a measure of dependence between random variables (Gretton et al., 2005). Unlike Pearson's correlation-coefficient which uses the values of the random variables, kernel-based dependence assesses the degree to which the similarity of example pairs, as defined by each kernel function, matches or aligns. In terms of distance-based kernel functions, the dependence could be posed as, "Do nearby examples, as defined by the first random variable, correspond to nearby examples in the second random variable?"

Consider the statistical alignment of two kernel functions. Let $z \in \mathcal{Z}$ denote a random variable and z' be an independent and identically distributed random variable. Let κ_1 and κ_2 be two kernel functions with implicit mappings $\phi_i : \mathcal{Z} \to \mathcal{H}_i$. A natural measure of similarity between these kernel functions is the expected value of their normalized inner product across pairs of realizations

$$A(\kappa_1, \kappa_2) = \frac{\mathrm{E}_{z,z'}[\kappa_1(z, z')\kappa_2(z, z')]}{\sqrt{\mathrm{E}_{z,z'}[\kappa_1^2(z, z')]\mathrm{E}_{z,z'}[\kappa_2^2(z, z')]}}. \tag{9}$$

Now, consider when $z = (x, y)$ represents a joint sample of $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ and κ_1, κ_2 only depend on x and y, respectively: $\kappa_1(z, z') = \kappa_x(x, x')$ and $\kappa_2(z, z') = \kappa_y(y, y')$. The marginal behavior of the kernels can be expressed in terms of their mapping functions:

$$\phi_1(z) = (\phi_x \otimes 1_y)(x, y), \text{ where } \phi_x : \mathcal{X} \to \mathcal{H}_x, \text{ and } \forall y \;\; 1_y(y) = 1 \tag{10}$$

$$\phi_2(z) = (1_x \otimes \phi_y)(x, y), \text{ where } \phi_y : \mathcal{Y} \to \mathcal{H}_y, \text{ and } \forall x \;\; 1_x(x) = 1. \tag{11}$$

$$\text{Then } A(\kappa_1, \kappa_2) = \frac{\mathrm{E}_{x,y}\mathrm{E}_{x',y'}[\kappa_x(x, x')\kappa_y(y, y')]}{\sqrt{\mathrm{E}_x\mathrm{E}_{x'}[\kappa_x^2(x, x')]\mathrm{E}_y\mathrm{E}_{y'}[\kappa_y^2(y, y')]}}. \tag{12}$$

is a measure of statistical dependence between x and y, since it is higher when similar pairs of one variable correspond to similar pairs in the other variable. However, the

measure performs poorly in practice without centering the kernels first (Cortes et al., 2012).

Centering plays the same role as removing the mean when computing the correlation coefficient between scalar-valued random variables. The centered kernel alignment is defined by Cortes et al. (2012) as

$$\rho(\kappa_1, \kappa_2) = A(\tilde{\kappa}_1, \tilde{\kappa}_2) \tag{13}$$

$$\begin{aligned}\tilde{\kappa}_i(z, z') &= \langle \tilde{\phi}_i(z), \tilde{\phi}_i(z') \rangle = \langle \phi_i(z) - \mathrm{E}_z[\phi_i(z)], \phi_i(z') - \mathrm{E}_{z'}[\phi_i(z')] \rangle \\ &= \kappa_i(z, z') - \mathrm{E}_{z'}[\kappa_i(z, z')] - \mathrm{E}_z[\kappa_i(z, z')] + \mathrm{E}_{z,z'}[\kappa_i(z, z')].\end{aligned} \tag{14}$$

Centering the mapping functions is key to a useful measure of dependence. The role of centering can be seen by expanding the numerator of the kernel target alignment in tensor product form:

$$\begin{aligned}\mathrm{E}_{z,z'}[\kappa_1(z, z')\kappa_2(z, z')] &= \mathrm{E}_{z,z'} \langle (\phi_1 \otimes \phi_2)(z, z), (\phi_1 \otimes \phi_2)(z', z') \rangle \\ &= \langle \mathrm{E}_z[(\phi_1 \circ \phi_2)(z)], \mathrm{E}_{z'}[(\phi_1 \circ \phi_2)(z')] \rangle \\ &= \| \mathrm{E}_z[(\phi_1 \circ \phi_2)(z)] \|_2^2\end{aligned} \tag{15}$$

Writing the original kernel in terms of the centered kernel (14) yields

$$\begin{aligned}\mathrm{E}_z(\phi_1 \circ \phi_2)(z) &= \mathrm{E}_z(\tilde{\phi}_1 + \mathrm{E}_{z'}[\phi_1(z')]) \circ (\tilde{\phi}_2 + \mathrm{E}_{z'}[\phi_2(z')])(z) \\ &= \mathrm{E}_z(\tilde{\phi}_1 \circ \tilde{\phi}_2)(z) + \mu_1 \circ \mu_2 + \mu_2 \circ \tilde{\phi}_1(z) + \mu_1 \circ \tilde{\phi}_2(z) \\ &= \mathrm{E}_z(\tilde{\phi}_1 \circ \tilde{\phi}_2)(z) + \mu_1 \circ \mu_2\end{aligned}$$

where $\mu_i = \mathrm{E}_z(\phi_i(z))$ and $\mathrm{E}_z(\tilde{\phi}_i(z)) = 0$. In terms of the marginal kernels

$$\mu_1 \circ \mu_2 = \mathrm{E}_{x,y}[(\phi_x \otimes 1_y)(x, y)] \circ \mathrm{E}_{x,y}[(1_x \otimes \phi_y)(x, y)] = (\mathrm{E}_x \phi_x(x)) \otimes (\mathrm{E}_y \phi_y(y)) = \mu_x \otimes \mu_y,$$

which is only a measure of the marginals—not of their joint distribution—thus its biases the norm in (15) regardless of the dependence between x and y.

Again if $z = (x, y)$ and $\kappa_1(z, z') = \kappa_x(x, x')$ and $\kappa_2(z, z') = \kappa_y(y, y')$, then $\rho(\kappa_1, \kappa_2) = \rho_{\kappa_x, \kappa_y}(x, y)$ is a measure of statistical dependence between x and y:

$$\rho_{\kappa_x, \kappa_y}(x, y) = \frac{\mathrm{E}_{x,y} \mathrm{E}_{x',y'}[\tilde{\kappa}_x(x, x')\tilde{\kappa}_y(y, y')]}{\sqrt{\mathrm{E}_x \mathrm{E}_{x'}[\tilde{\kappa}_x^2(x, x')] \mathrm{E}_y \mathrm{E}_{y'}[\tilde{\kappa}_y^2(y, y')]}}. \tag{16}$$

For positive-definite-symmetric kernels, $\rho_{\kappa_x, \kappa_y} \in [0, 1]$ (Cortes et al., 2012). Centered alignment is essentially a normalized version of the Hilbert-Schmidt Information Criterion (Gretton et al., 2005).

An empirical estimate of the centered alignment can be computed directly from the kernel matrices K and L where $[K]_{i,j} = \kappa_x(x_i, x_j)$ and $[L]_{i,j} = \kappa_y(y_i, y_j)$:

$$\hat{\rho}(K, L) = \frac{\langle \tilde{K}, \tilde{L} \rangle}{\sqrt{\langle \tilde{K}, \tilde{K} \rangle \langle \tilde{L}, \tilde{L} \rangle}} = \frac{\langle \tilde{K}, \tilde{L} \rangle}{\|\tilde{K}\|_2 \|\tilde{L}\|_2} \tag{17}$$

where $\tilde{K}$ and $\tilde{L}$ are the centered kernel matrices. The centered kernel is computed as

$$[\tilde{K}]_{i,j} = [K]_{i,j} - \frac{1}{n}\sum_{i=1}^{n}[K]_{i,j} - \frac{1}{n}\sum_{j=1}^{n}[K]_{i,j} + \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}[K]_{i,j}. \tag{18}$$

Using matrix multiplication, $\tilde{K} = HKH$, where $H = I - \frac{1}{n}\mathbf{1}\mathbf{1}^{\mathrm{T}}$ is the empirical centering matrix, I is the $n \times n$ identity matrix, and $\mathbf{1}$ is a vector of ones. The computational complexity of centered alignment between two $n \times n$ kernel matrices is $\mathcal{O}(n^2)$.

3.2 Metric Learning Optimization

Our objective for metric learning is to maximize the dependence between the neural data representation and the class label. Centered alignment is used to evaluate the dependence in terms of the kernel representations. The 0-1 kernel on the labels is fixed, and the parameters of a metric-based kernel defined in Section 2.2 are optimized in order to maximize the centered alignment.

For convenience, we use the logarithm of the centered alignment as the objective. With or without the logarithm the kernel-based objective is a nonlinear functions of the parameters, and we propose to use approximate inverse Hessian and stochastic gradient methods for optimization. We detail the gradients below.

First, we consider optimizing the sum and product kernels. As the product kernel (5) is the trivial case of the sum kernel (7), we consider only the optimization of the sum kernel parameters $\Theta = [\theta_i^j]_{i=1,j=1}^{P,Q}$ in the following problem:

$$\underset{\boldsymbol{\Theta}\geq 0}{\text{maximize}} \;\; \log(\rho_{\kappa_\Theta,\delta}(x,l)). \tag{19}$$

When the empirical estimate of centered alignment is substituted the explicit objective function is

$$f(\Theta) = \log(\hat{\rho}(K_\Theta, L)) = \log(\mathrm{tr}(K_\Theta HLH)) - \log(\sqrt{\mathrm{tr}(K_\Theta HK_\Theta H)}) + k \tag{20}$$

where K_Θ is the kernel matrix of the responses, L is the 0-1 kernel matrix of the labels, H is the centering matrix, and k is a constant that does not depend on Θ. The gradient of the objective function with respect to the kernel matrix is

$$G = \nabla_{K_\Theta} f(\Theta) = \frac{HLH}{\mathrm{tr}(K_\Theta HLH)} - \frac{HK_\Theta H}{\mathrm{tr}(K_\Theta HK_\Theta H)}. \tag{21}$$

The gradient kernel with respect to the kernel parameter θ_i^j is $\frac{\partial K_\Theta}{\partial \theta_i^j} = -K_{\theta^j} \circ D_i^{\circ\gamma}$. Then the gradient of the objective is

$$\frac{\partial f(\Theta)}{\partial \theta_i^j} = \mathrm{tr}\left((-K_{\theta^j} \circ D_i^{\circ\gamma})G\right). \tag{22}$$

The non-negativity constraint on Θ can be removed by performing the optimization in terms of u where $\theta_i^j = 10^{u_i^j}$. The gradients can be made in terms of unconstrained

optimization variables u_i by $\frac{\partial f(\Theta)}{\partial u_i^j} = \theta_i \log(10) \frac{\partial f(\Theta)}{\partial \theta_i^j}$. This yields an unconstrained optimization.

For the case of scalar-valued data we explore learning a Mahalanobis metric (6) using the logarithm of the empirical centered alignment as the objective:

$$\underset{A}{\text{maximize}} \;\; f(A) = \log(\hat{\rho}(K_A, L)) \tag{23}$$

The gradient of the objective function with respect to A is

$$\nabla_{\mathbf{A}} f(A) = -4X^{\mathrm{T}} \left((G \circ K_A) - \mathrm{diag}(\mathbf{1}^{\mathrm{T}}(G \circ K_A)) \right) XA \tag{24}$$

where X is a $n \times P$ matrix of the data, G is the gradient of the objective function with respect to the kernel matrix (21), and $\mathbf{1}$ is a vector of ones.

For the approximate inverse Hessian optimization we use `minFunc` (Schmidt, 2012), using the default limited memory BFGS update. For the sum and product kernels, prior to optimization the individual matrices $D_i^{\circ\gamma}$ are all normalized such that the average across all elements is 1. For the product kernel all the weights are initialized to be 10^{-3} and for the sum of product kernels they are uniformly distributed in $10^{-3} \pm 10^{-4}$. For the Mahalanobis distance, the optimization of A yields varying results depending on the initial value of A, but using the projection from Fisher discriminant analysis for initialization performs well in practice.

As an alternative optimization that can handle large sample sizes, we use a stochastic gradient over small batches. Specifically, we use a paradigm commonly used in feature selection: at each iteration, one example is sampled and then a pre-specified number of examples of the same class and from differing classes are sampled to form the batch (Kira and Rendell, 1992). For each batch, the weights are updated based on the gradient of the objective. Very small batches—even just four examples—are sufficient for learning the parameters of the product kernel, but to learn a Mahalanobis distance we found larges batches, in the hundreds, are necessary.

4 Benchmark Comparison

Using publicly available datasets, we compare centered alignment metric learning to optimize a weighted metric to a feature weighting method (Takeuchi and Sugiyama, 2011). The feature weighting is explicitly optimized to improve the k-nearest neighbor classification; this serves as a benchmark for centered alignment metric learning, which is not tuned to any particular classifier. The method was shown to consistently outperform other feature weighting methods. For a valid comparison the specifics of the benchmark comparison by Takeuchi and Sugiyama (2011) are replicated; we use the same UCI machine learning datasets (Bache and Lichman, 2013; Cortez et al., 2009; Little et al., 2007) and classification scenario (1/3 for training, 1/3 to choose k, number of nearest neighbors, through cross-validation, and 1/3 for testing). However, we increase the Monte Carlo divisions to 200 for statistical comparison. As a sanity check, Euclidean distance after normalizing the variance of the features is also used. We tested both the L-BFGS optimization and the mini-batch with 4 sample batches, 10,000 batches, and a step size of 0.01. We did not rerun the sequential quadratic program-based feature

weighting (Takeuchi and Sugiyama, 2011), but instead list the value they report for the mean error rate across 10 Monte Carlos divisions.

Table 1: k-nearest neighbor classification error (% incorrect) across 10 UCI datasets using different feature weightings. Columns denoted $|C|$, P, and n indicate the number of classes, features, and samples, respectively. The sequential quadratic program-based feature weighting (FW) (Takeuchi and Sugiyama, 2011) is compared to an unweighted normalized Euclidean distance (Euclid.) and centered alignment metric learning (CAML) to optimize a product kernel. The mini-batch approximation is indicated as (∼CAML). (**Bold** indicates best performing methods, which were not significantly different $p > 0.05$ for either a one-sample t-test versus FW, or a two-sample t-test.)

Dataset	$\|C\|$	P	n	FW	Euclid.	CAML	∼CAML
Pen-Based Recog.	10	16	10992	1.1	**1.0±0.2**	N/A	1.2±0.2
Breast Wisc. (Diag.)	2	30	569	**4.0**	4.8±1.7	4.4±1.5	4.3±1.4
Page Blocks	5	10	5473	4.6	**4.1±0.5**	4.6±0.5	4.3±0.5
Image Segmentation	7	18	2310	5.2	6.2±1.0	**3.3±0.8**	4.6±0.8
Ionosphere	2	33	351	12.2	16.3±3.9	**10.7±4.9**	13.7±3.5
Parkinsons	2	22	195	**10.2**	12.1±4.3	13.6±4.7	11.5±3.9
Spambase	2	57	4601	**10.4**	11.0±0.9	14.6±4.7	**10.4±0.8**
Waveform (ver. 1)	3	21	5000	18.4	19.0±0.9	**17.9**±0.9	18.5±0.8
Connectionist (Sonar)	2	60	208	22.1	**20.8±5.4**	27.5±4.9	22.4±5.4
Wine Quality	7	11	6497	46.3	46.3±1.0	48.9±1.2	**46.0±1.0**

The results are displayed in Table 1. On these small scale problems—maximum dimensions is 57—none of the compared methods consistently outperforms the best. Considering the best of the two proposed optimization methods, centered alignment metric learning performs best on half of the datasets.

5 Data collection

All animal procedures were approved by the SUNY Downstate Medical Center IACUC and conformed to National Institutes of Health guidelines. Cortical local field potentials and action potentials were recorded during natural tactile stimulation of forepaw digits and palm of 4 female Long-Evans rats under anesthesia.[1] After induction using isoflurane, urethane was used to maintain anesthetic depth. A 32-channel microelectrode array (Blackrock Microsystems) was inserted into the hand region of the primary somatosensory cortex (S1). The array was arranged in a 6×6 grid (excluding the four corners) with 400 μm spacing between neighboring electrodes. Another array was inserted into the VPL region of the thalamus, but the signals are not used here.

Using a motorized probe, the right forepaw was touched 225 times at up to 9 sites—4 digits and 5 sites on the palm. For each touch site, the probe was positioned 4 mm above the surface of the skin and momentarily pressed down for 150 ms, as seen in Figure 1; this was repeated 25 times at random intervals. The 4 datasets had 3, 8, 9, and 9 touch sites resulting in 75, 200, 225, and 225 samples, respectively.

The LFPs were band-pass filtered with cutoffs (5 Hz, 300 Hz) and sampled at a rate of 1220.7 Hz. Then the LFPs were digitally filtered using a 3rd-order Butterworth high-pass filter with cutoff of 4 Hz and notch filters at 60 Hz and harmonics. For analysis, the neural response in a 270ms window following each touch onset was used, which corresponds to 330 discrete time samples. For 32 channels this results in $330 \cdot 32 = 10,560$ dimensions.

Across the 4 datasets, automatic spike-sorting selected 95, 64, 64, and 38 multi-neuron units from the 32 channels. Of these, only 68, 62, 36, and 24 units were used, whose average firing rate was below 30 Hz in the 270 ms window following touch onset.

Figure 1: Experimental setup showing motorized lever touching digit 1 on the forepaw.

6 Results

We explored centered alignment metric learning (CAML) for both spike trains and local field potentials (LFPs) using the cases listed in Section 2.3. For LFPs and binned spike trains we compared with multi-class Fisher discriminant analysis (FDA) (Fukunaga, 1990) and large-margin nearest neighbor (LMNN) (Weinberger et al., 2006; Weinberger

[1] A subset of this data has been analyzed previously using other methods (Brockmeier et al., 2013).

and Saul, 2009). For the linear projections, PCA was used to reduce the dimensionality to 1/2 of the number of samples in the dataset. The FDA solution is the set of eigenvectors corresponding to a generalized eigenvalue problem. The dimensions can be chosen as the maximum number of non-zero eigenvalues, which is one less than the number of classes (Fukunaga, 1990). The FDA solution was used as the initial projection for LMNN and CAML Mahalanobis metric. An efficient MATLAB implementation of LMNN is publicly available, and besides the initialization (which greatly increased the performance) default parameters were used.

To compare classification performance, 20 Monte Carlo divisions of the datasets into training and testing sets were made. For training, 2/3 of the samples in each class were used, the remainder of the samples were used in testing. On each Monte Carlo run, the metrics were optimized on the training set. Testing set samples were labeled by either a one-nearest-neighbor (1-NN) or a support vector machine (SVM) classifier. SVM training and testing was performed using the `libsvm` (ver. 3.17) (Chang and Lin, 2011) implementation with the user-provided kernel matrix. The regularization parameter was chosen through 5-fold cross-validation. For CAML the kernel is directly optimized as part of the metric learning, but for FDA, LMNN, and the unweighted metrics a Gaussian kernel was used with the kernel size chosen from a discrete set using 5-fold cross-validation.

The set of the highest performing methods for each dataset was found by selecting the best performing method and finding those that were not significantly different using a two-sample Welch test with significance of 0.05.

6.1 Decoding Touch Location from Spike Trains

Multiunit spike-train metrics using the single-unit Victor-Purpura (VP) and kernel-based (mCI) metrics were optimized for touch location classification. For each unit the distance is computed with different values for the temporal precision value q (higher values of q require more precise alignment): for the Victor-Purpura distance the set (0.01, 0.1, 1.0) s^{-1} was used, and for the spike train kernel-based metrics (mCI) the set (10^{-9}, 0.01, 0.1, 1, 10, 100) s^{-1} was used. For the Victor-Purpura distance, the $\mathcal{L}_2$ version (Dubbs et al., 2010) was used.[2] The classification rates for the weighted spike-train metrics are in Table 2. With the CAML-optimized product kernel the average classification rate increased by at least 8 percentage points for both metrics and both classifiers. For the sum kernel with $Q = 5$ the accuracy was further increased.

For binned spike trains a Mahalanobis metric was optimized using FDA, CAML, and LMNN. The results across a range of different bin sizes are shown in Figure 2. On three datasets the best binned metrics performed worse than the optimized spike-train metrics. For each dataset and method the performance using the bin size with the highest average accuracy is shown in Table 3. On three of the datasets the Mahalanobis metric optimized with CAML tied or outperformed the FDA solution, and on all datasets using LMNN decreased performance.

We used classical multidimensional scaling (MDS) (Torgerson, 1952) and t-distributed

[2] Experiments with the original $\mathcal{L}_1$ version (Victor and Purpura, 1996) with a Laplacian kernel were also performed, but there was no significant difference.

Table 2: Spike train touch site classification accuracy (% correct) across 4 datasets using Victor-Purpura (VP) or kernel-based (mCI) metrics in unweighted combinations versus using centered alignment metric learning (CAML) to optimize a product kernel (θ) or sum of weighted product kernels (Θ) with $Q = 5$. Nearest-neighbor (1-NN) and support vector machine (SVM) were used as classifiers. (**Bold** indicates methods with highest accuracy with or without binning see Table 3.)

VP unweighted		mCI unweighted		CAML VP θ	CAML VP θ	CAML VP Θ	CAML mCI θ	CAML mCI θ	CAML mCI Θ
1-NN	SVM	1-NN	SVM	1-NN	SVM	SVM	1-NN	SVM	SVM
53±9	69±9	60±9	59±8	86±6	87±5	85±9	85±4	**90±6**	**92±5**
35±4	80±5	70±5	78±5	77±5	87±5	**91±4**	78±4	87±5	**89±3**
28±5	50±4	43±4	50±6	44±6	53±4	**59±5**	48±5	58±6	**61±4**
22±4	28±6	25±4	27±5	22±5	**38±5**	**38±5**	22±4	29±9	34±4
34.6	56.9	49.2	53.7	57.2	66.1	68.3	58.5	66.0	69.0

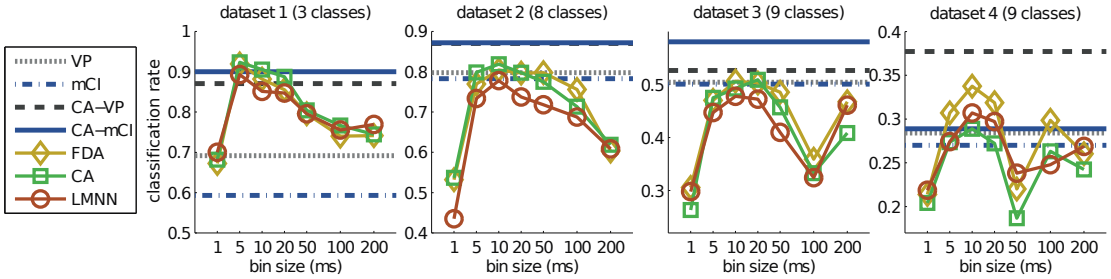

Figure 2: SVM-based classification rate using multi-unit spike train metrics and binned spike count metrics with varying binsizes: unweighted Victor-Purpura (VP) and kernel-based (mCI) metrics, centered alignment metric learning optimized spike train metrics (CA-VP, CA-mCI), and Mahalanobis metrics on binned spike trains optimized with Fisher discriminant analysis (FDA), centered alignment (CA), and large margin nearest neighbor (LMNN).

stochastic neighborhood embedding (t-SNE) (van der Maaten and Hinton, 2008) to find a two-dimensional embedding of the distance matrices before and after training the metric. The embedding is formed without knowledge of the class labels. From Figure 3 it is clear that metric learning with the product kernel increases distances among the different classes while decreasing the distances among samples within the same class. In Figure 4(A), we show how the optimized spike-train metric can be used to identify both the temporal precision and spiking units most useful for the decoding task.

6.2 Decoding Touch Location from LFPs

The classification rates for learning spatiotemporal metrics on LFP are tabulated in Table 4. Using CAML to optimize just a single temporal weighting improves the accuracy

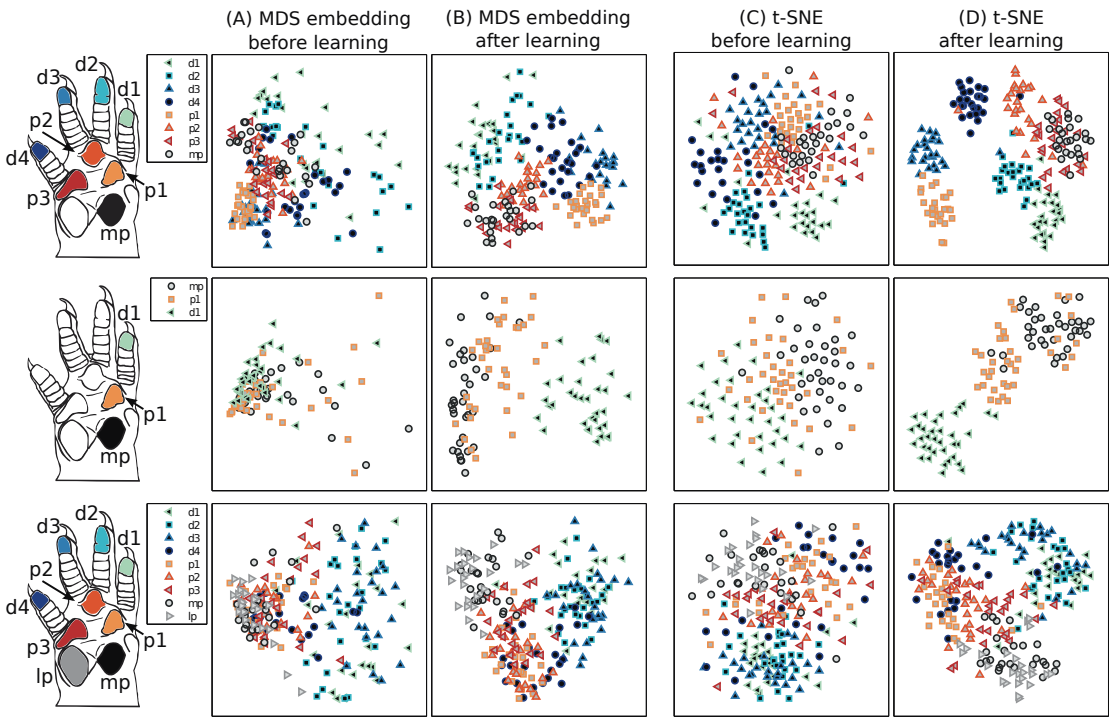

Figure 3: Comparison of metric-based dimensionality reduction before and after using centered alignment metric learning (CAML) to optimize a weighted combination of Victor-Purpura spike train distances. In (A) and (B) a two-dimensional embedding is formed using multidimensional scaling (MDS) before and after learning. In (C) and (D) t-distributed stochastic neighborhood embedding (t-SNE) is used to form a nonlinear embedding where the algorithm's perplexity parameter was fixed at 10.

by 21 and 14.7 percentage points for 1-NN and SVM, respectively. Using a sum kernel composed of $Q = 5$ product kernels further increased the performance by 2.2 and 4 percentage points. The optimized weights for a single product kernel are shown in Figure 4(B). Overall, using FDA to optimize a linear projection was able to achieve the highest classification rates with average improvement over Euclidean distance by 33.5 and 23.4 percentage points for 1-NN and SVM.

Finally, a multiscale metric was learned as the weighted combination of the optimized spike distance and optimized LFP distance. On these datasets the combination of spiking and LFPs did not increase the classification rate versus using only LFPs, and the weight assigned to the spiking metric was insignificant compared to the weight assigned to the LFP metric. The average classification accuracy across the datasets was slightly lower than using just the LFP metric.

Table 3: Binned spike train touch site classification accuracy (% correct) across 4 datasets using Euclidean and Mahalanobis-based metrics (parametrized by matrix A) optimized using Fisher discriminant analysis (FDA), centered alignment (CA), and large margin nearest neighbor (LMNN). For each dataset and method the binsize with the maximum performance was selected. (**Bold** indicates highest performing methods for each dataset with or without binning see Table 2.)

Euclidean		A-FDA		A-CA		A-LMNN	
1-NN	SVM	1-NN	SVM	1-NN	SVM	1-NN	SVM
58±10	73±9	**91±8**	**92±14**	**92±9**	**92±10**	**91±12**	**89±14**
66±6	76±7	80±8	80±9	82±6	82±7	77±8	78±10
42±5	46±8	47±6	51±6	51±6	51±7	46±7	48±10
24±5	28±6	30±6	34±7	29±6	29±6	29±6	31±6
47.4	55.6	61.9	64.2	63.5	63.5	60.7	61.4

7 Discussion

7.1 Metric Learning for Neural Decoding

From the results it is clear that metric learning achieves three goals: increases the decoding accuracy, identifies important dimensions of the neural response, and improves the ability of manifold learning techniques to visualize the data in a low-dimensional space.

For spike trains, the average performance of the optimized multi-unit spike train metrics exceeded those based on binning. To our knowledge this is the first work on optimizing a multi-neuron metric that is non-parametric and does not require binning. In the framework of the kernel-based dependence measure, this optimization explicitly optimizes the contribution of each dimension using tensor product kernels for multi-neuron spike trains.

On all datasets the performance from using the unweighted multi-neuron spike train metrics was lower than using the optimized Mahalanobis metrics on the binned representation. In essence, a simple linear projection of binned spike trains performs better than binless metrics that are not optimized. The precision offered by binless methods is only realized after optimization. This highlights the importance of metric learning versus naively combining the single-unit metrics.

FDA achieved the best performance for this static decoding task using the time-locked evoked LFPs. FDA is well-suited for this setting since the class conditional LFP responses are approximately normally distributed—an underlying assumption for FDA. In addition, the FDA solution is also the fastest solution; consequently, FDA should always be the baseline for discrete decoding of sensory evoked LFPs. Alternatively, for binned spikes using CAML to further optimize the FDA projection marginally increased the classification performance. Overall, FDA and CAML outperformed LMNN in optimizing a Mahalanobis metric.

One drawback of the Mahalanobis metric is the ability to analyze the projection

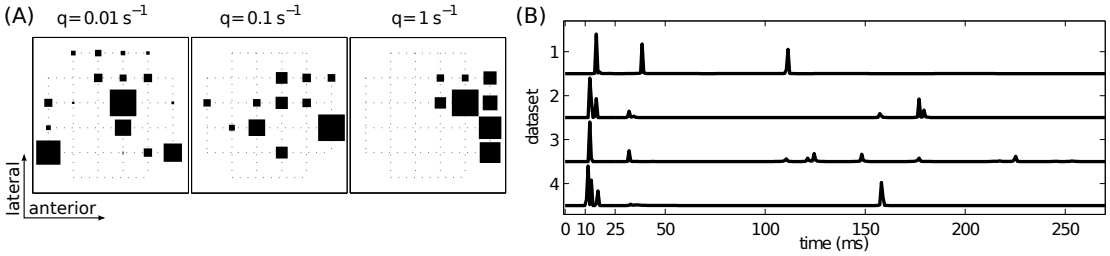

Figure 4: (A) Learned weights for the optimized Victor-Purpura spike-train metric (CAML-VP) shown across the array as a Hinton diagram—the size of each square is relative to the maximum weight of all units on the channel. Each subplot shows the weights for a different choice of the temporal precision value q; the weights for all temporal precision values are learned at the same time for dataset 2. (B) Learned weighting for each time lag across all datasets for the local field potential metric trained using centered alignment metric learning with the product kernel (θ-CAML).

matrices themselves, i.e, it is difficult to match and compare linear projections learned across multiple subjects or tasks, especially for high-rank projections. In this case using a weighted metric, which has lower accuracy but far fewer parameters, is more easily interpretable. From Figure 4B it is clear that the weighted metrics can be used to identify dimensions, in this case time lags, that are useful for discrimination. In addition, it appears that the optimization leads to a very sparse set of weights.

In terms of neural decoding, we compared the classification rate, as a proxy for the information content, of the neural responses. We have also highlighted how changing the underlying metric of the neural response space can improve the visualization results from unsupervised manifold learning algorithms. Indeed from Figure 3 a user can immediately judge which classes are more similar or indistinguishable. The non-linear embeddings preserve some features of the stimulus space's topology, e.g., the separation between digit responses and palm responses in dataset 3 and the preservation of the relative arrangement of the three touch sites in dataset 2.

7.2 Metric Learning for Neural Encoding

We have concentrated on the problem of neural decoding, but the proposed algorithms are also applicable to the neural encoding problem, wherein, the role of the stimulus and neural response are reversed. More specifically, for neural encoding, the metric on the neural response is fixed, the neural activity, e.g., the spiking of a single neuron, is treated as the target or label variable and a metric on the stimulus is adjusted. For instance, if the neuron is assumed to be a simple cell with a linear receptive field, then learning the receptive field is equivalent to learning a Mahalanobis distance on the stimulus space.

The ideas that have been developed for metric-learning/supervised dimensionality reduction in the machine learning community are fundamentally similar to the algorithms for inferring the linear receptive fields of neurons in the computational neuroscience community, but the nomenclature and domain has differentiated them. Re-

Table 4: LFP classification accuracy (% correct) across 4 datasets using centered alignment metric learning (CAML) to optimize a single temporal weight vector θ or a sum kernel with multiple temporal weightings Θ, versus Mahalanobis-based metrics (parametrized by a matrix A) optimized using Fisher discriminant analysis (FDA), centered alignment (CA), and large margin nearest neighbor (LMNN). (**Bold** indicates highest performing methods for each dataset.)

Euclidean	θ-CAML	Θ-CAML	A-CA	A-FDA	A-LMNN
			1-NN		
85±4.6	92±3.6	94±4.0	**98±2.4**	**98±1.9**	97±1.7
60±4.5	78±4.4	82±4.4	95±2.6	**97±1.7**	94±4.4
45±3.5	74±5.0	78±4.4	**91±5.0**	**91±3.5**	88±4.4
49±4.1	78±4.3	78±5.2	**85±3.4**	**86±4.0**	80±5.9
59.6	80.6	82.8	92.4	93.1	89.7
			SVM		
89±4.4	94±3.7	97±2.2	**98±2.0**	**98±2.0**	97±2.4
75±5.0	84±3.0	89±2.4	95±2.6	**97±1.9**	94±5.0
61±5.4	79±4.9	83±4.0	**91±4.7**	**92±3.5**	87±2.9
54±5.1	81±4.2	**85±3.8**	**86±3.8**	**86±4.5**	80±5.9
69.8	84.5	88.5	92.4	93.2	89.3

cently, researchers have begun to bridge this gap using kernel-based measures of dependence (Sinz et al., 2013). To further highlight this connection, we replicated the experiments by Sharpee et al. (2004), but instead of using the maximally informative directions algorithm we used centered alignment metric learning, with the mini-batch optimization. To save space the experimental detail and results are posted online.[3]

Interestingly, most neural encoding models have concentrated on linear projections corresponding to Mahalanobis-based distances, whereas recent work has shown that the stimulus metric corresponding to a neural population can be highly non-Euclidean Tkačik et al. (2013). Thus, future work can investigate how non-Euclidean metrics can be learned. Additionally, the joint optimization of the metrics on both the neural response and the stimulus is worth investigating in future work.

7.3 Kernel-based Metric Learning

In the kernel-learning framework, we have proposed to use a weighted product kernel, where adjusting the weights changes the underlying kernel and associated Hilbert space. This leads to non-convex optimization, which we solve using first-order methods. In addition, the sum of weighted product kernels uses a different set of weights for each product kernel and achieves increased performance. This formulation is distinct from the multiple kernel learning Lanckriet et al. (2004); Cortes et al. (2012); Yamada et al. (2013), where there is an explicit weight associated with each kernel in the sum

[3] http://cnel.ufl.edu/%7Eajbrockmeier/metric/

and each summand kernel is chosen a priori (i.e., in the case of Gaussian kernel, the kernel size is not optimized and must be preselected). The main benefit of the multiple kernel learning is that a convex optimization problem can be posed to optimize the weights. Alternatively, the weighted product kernels and sum of weighted product kernels constitutes a much richer family of kernel functions than the weighted sum of kernels. We only need to select the number of summand kernels; fully exploring how to chose this number is left for future work.

Linear projections and weighted metrics are two special cases of metric learning that have received the most study. Indeed, the weighted combination of metrics was used in some of the earliest work on metric learning (Lowe, 1995). We have gone beyond this by using a sum of weighted product kernels, which computes distances in the Hilbert space that correspond to nonlinear transformations of the data samples. The sum of weighed product kernels still has interpretable parameters, quite unlike kernelized projections (Baudat and Anouar, 2000), where the transformations are only defined in terms of the samples instead of the original dimensions.

Conclusion

We have covered a class of kernels applicable to metric learning and have proposed to use a kernel-based dependence measure to train linear projections, weighted combinations of metrics in product kernels, and nonlinear combinations of metrics using the sum of weighted product kernels. These metrics were optimized on a neural dataset consisting of both spike trains and local field potentials, for which metric learning improves both nearest neighbor and SVM classification accuracy over unweighted alternatives. Within the proposed framework, the optimized multiunit spike train metrics, which avoid binning, outperform both unweighted multiunit metrics and metrics optimized for the binned spike trains. In addition, metric learning improves the quality of the visualizations obtained via nonlinear dimensionality reduction; this is useful for analyzing the relationship between high-dimensional neural data and target variables. The optimized weights themselves indicate the relative relevancy of different dimensions of the neural response. This can be used to explore how the weights of specific channels or neurons change for different tasks. Overall, optimizing metrics is a worthwhile approach for investigating neural representations.

Acknowledgments

This work was supported in part by DARPA Contract N66001-10-C-2008. We want to thank Memming Park, Luis Giraldo Sanchez, and Sohan Seth for their helpful discussions and insight on topics that lead to this work. We would like to thank the reviewers for suggestions that have improved this work.

References

Aronov, D. (2003). Fast algorithm for the metric-space analysis of simultaneous responses of multiple single neurons. *Journal of Neuroscience Methods*, 124(2):175–179.

Bach, F. R. and Jordan, M. I. (2003). Kernel independent component analysis. *The Journal of Machine Learning Research*, 3:1–48.

Bache, K. and Lichman, M. (2013). UCI machine learning repository.

Baudat, G. and Anouar, F. (2000). Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12(10):2385–2404.

Brockmeier, A. J., Kriminger, E., Sanchez, J. C., and Principe, J. C. (2011). Latent state visualization of neural firing rates. In *Neural Engineering (NER), 2011 5th International IEEE/EMBS Conference on*, pages 144–147.

Brockmeier, A. J., Park, I., Mahmoudi, B., Sanchez, J. C., and Principe, J. C. (2010). Spatio-temporal clustering of firing rates for neural state estimation. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 6023–6026.

Brockmeier, A. J., Sanchez Giraldo, L. G., Emigh, M. S., Bae, J., Choi, J. S., Francis, J. T., and Principe, J. C. (2013). Information-theoretic metric learning: 2–D linear projections of neural data for visualization. In *Engineering in Medicine and Biology Society (EMBC), 2013 Annual International Conference of the IEEE*.

Broome, B. M., Jayaraman, V., and Laurent, G. (2006). Encoding and decoding of overlapping odor sequences. *Neuron*, 51(4):467–482.

Bunte, K., Biehl, M., and Hammer, B. (2012). A general framework for dimensionality-reducing data visualization mapping. *Neural Computation*, 24(3):771–804.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Churchland, M. M., Yu, B. M., Cunningham, J. P., Sugrue, L. P., Cohen, M. R., Corrado, G. S., Newsome, W. T., Clark, A. M., Hosseini, P., Scott, B. B., Bradley, D. C., Smith, M. A., Kohn, A., Movshon, J. A., Armstrong, K. M., Moore, T., Chang, S. W., Snyder, L. H., Lisberger, S. G., Priebe, N. J., Finn, I. M., Ferster, D., Ryu, S. I., Santhanam, G., Sahani, M., and Shenoy, K. V. (2010). Stimulus onset quenches neural variability: a widespread cortical phenomenon. *Nature Neuroscience*, 13(3):369–378.

Cortes, C., Mohri, M., and Rostamizadeh, A. (2012). Algorithms for learning kernels based on centered alignment. *The Journal of Machine Learning Research*, 13:795–828.

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., and Reis, J. (2009). Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553.

Cowley, B., Kaufman, M., Churchland, M., Ryu, S., Shenoy, K., and Yu, B. (2012). Datahigh: Graphical user interface for visualizing and interacting with high-dimensional neural activity. In *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pages 4607–4610.

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., and Kandola, J. (2002). On kernel-target alignment. In Dietterich, T., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA. MIT Press.

Dubbs, A. J., Seiler, B. A., and Magnasco, M. O. (2010). A fast $\mathcal{L}_p$ spike alignment metric. *Neural Computation*, 22(11):2785–2808.

Fukumizu, K., Bach, F. R., and Jordan, M. I. (2004). Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces. *The Journal of Machine Learning Research*, 5:73–99.

Fukunaga, K. (1990). *Introduction to statistical pattern recognition*. Academic Press.

Gretton, A., Bousquet, O., Smola, A., and Schölkopf, B. (2005). Measuring statistical dependence with hilbert-schmidt norms. In *Algorithmic learning theory*, pages 63–77. Springer.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182.

Horn, R. A. (1967). On infinitely divisible matrices, kernels, and functions. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 8(3):219–230.

Houghton, C. and Sen, K. (2008). A new multineuron spike train metric. *Neural Computation*, 20(6):1495–1511.

Kemere, C., Santhanam, G., Yu, B., Afshar, A., Ryu, S., Meng, T., and Shenoy, K. (2008). Detecting neural-state transitions using hidden Markov models for motor cortical prostheses. *Journal of Neurophysiology*, 100(4):2441.

Kira, K. and Rendell, L. A. (1992). The feature selection problem: Traditional methods and a new algorithm. In *AAAI*, pages 129–134.

Lanckriet, G. R., Cristianini, N., Bartlett, P., Ghaoui, L. E., and Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming. *The Journal of Machine Learning Research*, 5:27–72.

Little, M. A., McSharry, P. E., Roberts, S. J., Costello, D. A., Moroz, I. M., et al. (2007). Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection. *BioMedical Engineering OnLine*, 6(1):23.

Lowe, D. G. (1995). Similarity metric learning for a variable-kernel classifier. *Neural Computation*, 7(1):72–85.

Navot, A., Shpigelman, L., Tishby, N., and Vaadia, E. (2006). Nearest neighbor based feature selection for regression and its application to neural activity. In Weiss, Y., Schölkopf, B., and Platt, J., editors, *Advances in Neural Information Processing Systems 18*, pages 995–1002. MIT Press, Cambridge, MA.

Paiva, A. R., Park, I., and Príncipe, J. C. (2009). A reproducing kernel Hilbert space framework for spike train signal processing. *Neural Computation*, 21(2):424–449.

Park, I. M., Seth, S., Paiva, A., Li, L., and Principe, J. (2013). Kernel methods on spike train space for neuroscience: A tutorial. *Signal Processing Magazine, IEEE*, 30(4):149–160.

Park, I. M., Seth, S., Rao, M., and Príncipe, J. C. (2012). Strictly positive-definite spike train kernels for point-process divergences. *Neural Computation*, 24(8):2223–2250.

Peng, H., Long, F., and Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1226–1238.

Petreska, B., Yu, B. M., Cunningham, J. P., Santhanam, G., Ryu, S. I., Shenoy, K. V., and Sahani, M. (2011). Dynamical segmentation of single trials from population neural data. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 24*, pages 756–764.

Radons, G., Becker, J., Dülfer, B., and Krüger, J. (1994). Analysis, classification, and coding of multielectrode spike trains with hidden Markov models. *Biological Cybernetics*, 71:359–373.

Roweis, S. and Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323.

Sammon, Jr., J. W. (1969). A nonlinear mapping for data structure analysis. *Computers, IEEE Transactions on*, C-18(5):401–409.

Sanchez Giraldo, L. G. and Principe, J. C. (2013). Information theoretic learning with infinitely divisible kernels. In *International Conference on Learning Representations*.

Schmidt, M. (2012). minFunc. Software available at `http://www.di.ens.fr/~mschmidt/Software/minFunc.html`.

Schoenberg, I. J. (1938). Metric spaces and positive definite functions. *Transactions of the American Mathematical Society*, 44(3):522–536.

Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319.

Seidemann, E., Meilijson, I., Abeles, M., Bergman, H., and Vaadia, E. (1996). Simultaneously recorded single units in the frontal cortex go through sequences of discrete and stable states in monkeys performing a delayed localization task. *Journal of Neuroscience*, 16(2):752.

Sharpee, T., Rust, N. C., and Bialek, W. (2004). Analyzing neural responses to natural signals: maximally informative dimensions. *Neural computation*, 16(2):223–250.

Sinz, F., Stockl, A., Grewe, J., and Benda, J. (2013). Least informative dimensions. In *Advances in Neural Information Processing Systems 26*, pages 413–421.

Song, L., Bedo, J., Borgwardt, K. M., Gretton, A., and Smola, A. (2007). Gene selection via the BAHSIC family of algorithms. *Bioinformatics*, 23(13):i490–i498.

Song, L., Smola, A., Gretton, A., Bedo, J., and Borgwardt, K. (2012). Feature selection via dependence maximization. *The Journal of Machine Learning Research*, 13:1393–1434.

Stopfer, M., Jayaraman, V., and Laurent, G. (2003). Intensity versus identity coding in an olfactory system. *Neuron*, 39(6):991–1004.

Sugiyama, M. (2007). Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. *The Journal of Machine Learning Research*, 8:1027–1061.

Takeuchi, I. and Sugiyama, M. (2011). Target neighbor consistent feature weighting for nearest neighbor classification. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 24*, pages 576–584.

Tenenbaum, J. B., De Silva, V., and Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.

Tkačik, G., Granot-Atedgi, E., Segev, R., and Schneidman, E. (2013). Retinal metric: a stimulus distance measure derived from population neural responses. *Physical review letters*, 110(5):058104.

Torgerson, W. S. (1952). Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419.

van der Maaten, L. and Hinton, G. (2008). Visualizing Data using t-SNE. *The Journal of Machine Learning Research*, 9:2579–2605.

Van Rossum, M. C. W. (2001). A novel spike distance. *Neural Computation*, 13:751–763.

Victor, J. D. and Purpura, K. P. (1996). Nature and precision of temporal coding in visual cortex: a metric-space analysis. *Journal of Neurophysiology*, 76(2):1310–1326.

Weinberger, K., Blitzer, J., and Saul, L. (2006). Distance metric learning for large margin nearest neighbor classification. In Weiss, Y., Schölkopf, B., and Platt, J., editors, *Advances in Neural Information Processing Systems 18*, pages 1473–1480. MIT Press, Cambridge, MA.

Weinberger, K. Q. and Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research*, 10:207–244.

Weinberger, K. Q. and Tesauro, G. (2007). Metric learning for kernel regression. In *International Conference on Artificial Intelligence and Statistics*, pages 612–619.

Xing, E. P., Ng, A. Y., Jordan, M. I., and Russell, S. (2003). Distance metric learning with application to clustering with side-information. In S. Becker, S. T. and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 505–512. MIT Press, Cambridge, MA.

Yamada, M., Jitkrittum, W., Sigal, L., Xing, E. P., and Sugiyama, M. (2013). High-dimensional feature selection by feature-wise kernelized lasso. *Neural Computation*, 26(1):185–207.

Yu, B. M., Cunningham, J. P., Santhanam, G., Ryu, S. I., Shenoy, K. V., and Sahani, M. (2009a). Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 1881–1888.

Yu, B. M., Cunningham, J. P., Santhanam, G., Ryu, S. I., Shenoy, K. V., and Sahani, M. (2009b). Gaussian-Process Factor Analysis for Low-Dimensional Single-Trial Analysis of Neural Population Activity. *Journal of Neurophysiology*, 102(1):614–635.