Hyperparameter Learning for Conditional Mean Embeddings with Rademacher Complexity Bounds

Anonymous Author 1
Unknown Institution 1

Anonymous Author 2
Unknown Institution 2

Anonymous Author 3
Unknown Institution 3

Abstract

Conditional mean embeddings are nonparametric models that encode conditional expectations in a reproducing kernel Hilbert space, creating a flexible and powerful framework for probabilistic inference. Their performance is greatly dependent on the choice of kernel hyperparameters and regularization parameter. However, current approaches for hyperparameter tuning predominantly rely on cross validation and distance based heuristics. For conditional mean embeddings with categorical targets and arbitrary inputs, we propose a hyperparameter learning framework based on Rademacher complexity bounds to prevent overfitting by balancing data fit against model complexity. The approach only requires batch updates, allowing scalable kernel learning without kernel approximations. Experiments demonstrate that our framework outperforms competing methods, and can be further extended to incorporate and learn neural network weights improving generalization.

1 Introduction

Conditional mean embeddings (CMEs) are attractive because they encode conditional expectations in a reproducing kernel Hilbert space (RKHS), bypassing the need for a parametrized distribution [Song et al., 2013]. They are part of a broader class of models known as kernel embeddings, where nonparametric probabilistic inference can be carried out entirely within the RKHS because difficult marginalization integrals become simple linear algebra [Muandet et al., 2016]. This very general framework is core to modern kernel probabilistic methods, including kernel two-sample testing [Gretton et al., 2007], kernel Bayesian inference [Fukumizu et al., 2013], density estimation [Kanagawa and Fukumizu, 2014, Song et al., 2008], component analysis [Muandet et al., 2013], dimensionality reduction [Fukumizu et al., 2004], feature discovery [Jitkrittum et al., 2016], and state space filtering [Kanagawa et al., 2016].

Nevertheless, like most kernel based models, their performance is highly dependent on the hyperparameters chosen. For these models, the model selection process usually begins by selecting a kernel, whose parameters become part of the model hyperparameters, which may further include noise or regularization parameters. Given a set of hyperparameters, training is performed by solving either a convex optimization problem, such as for support vector machines (SVMs) [Schölkopf and Smola, 2002], or a set of linear equations, such as for Gaussian processes (GPs) [Rasmussen and Williams, 2006], regularized least squares classifiers (RLSCs) [Rifkin et al., 2003], and CMEs. Unfortunately, hyperparameter tuning is not straight forward, and often cross validation [Song et al., 2013] or median length heuristics [Muandet et al., 2016] remain as the primary approaches for this task.

One notable success story in this domain are GPs, which utilizes the marginal likelihood for hyperparameter learning. The marginal likelihood arises from its Bayesian formulation, and exhibits certain desirable properties – in particular, the ability to automatically balance between data fit and model complexity. On the other hand, CMEs are not necessarily Bayesian, and hence they do not benefit from a natural marginal likelihood formulation, yet such a balance is critical when generalizing the model beyond known examples.

Can we formulate a learning objective for CMEs to balance data fit and model complexity, similar to the marginal likelihood of GPs? For CMEs with categorical targets and arbitrary input, we present such a learning objective as our main contribution. In particular, we: (1) derive a measure $r(\theta, \lambda)$ to measure the model complexity of a CME based on the Rademacher complexity of a relevant class of CMEs, (2) propose a novel learning

Preliminary work. Under review by AISTATS 2018. Do not distribute.

objective based on this complexity measure to control generalization risk by balancing data fit against model complexity, and (3) propose a scalable hyperparameter learning algorithm under this objective using batch updates. We show that this learning objective produces CMEs that generalize better than that learned from cross validation, empirical risk minimization (ERM), or median length heuristics on standard benchmarks, and apply such an algorithm to incorporate and learn neural network weights improving generalization accuracy.

2 Background and Related Work

2.1 Conditional Mean Embeddings

To construct a conditional embedding operator $\mathcal{U}_{Y|X}$ corresponding to the distribution $\mathbb{P}_{Y|X}$, where $X : \Omega \to \mathcal{X}$ and $Y : \Omega \to \mathcal{Y}$ are measurable random variables, we first choose a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ for the input space $\mathcal{X}$ and another kernel $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ for the output space $\mathcal{Y}$. These kernels k and l each describe how similarity is measured within their respective domains $\mathcal{X}$ and $\mathcal{Y}$, and are symmetric positive definite such that they uniquely define the RKHS $\mathcal{H}_k$ and $\mathcal{H}_l$. The conditional embedding operator $\mathcal{U}_{Y|X}$ is then the operator $\mathcal{U} : \mathcal{H}_k \to \mathcal{H}_l$ for which $\mu_{Y|X=x} = \mathcal{U}k(x, \cdot)$, where $\mu_{Y|X=x} := \mathbb{E}[l(Y, \cdot)|X = x]$ is the CME [Song et al., 2009]. In this sense, it sweeps out a family of conditional embeddings $\mu_{Y|X=x}$ in $\mathcal{H}_l$, each indexed by the input variable $x \in \mathcal{X}$. We then define cross covariance operators $C_{YX} := \mathbb{E}[l(Y, \cdot) \otimes k(X, \cdot)] : \mathcal{H}_k \to \mathcal{H}_l$ and $C_{XX} := \mathbb{E}[k(X, \cdot) \otimes k(X, \cdot)] : \mathcal{H}_k \to \mathcal{H}_k$. Alternatively, they can be seen as elements within the tensor product space $C_{YX} \in \mathcal{H}_l \otimes \mathcal{H}_k$ and $C_{XX} \in \mathcal{H}_k \otimes \mathcal{H}_k$.

Under the assumption that $k(x, \cdot) \in \text{image}(C_{XX})$, it can be shown that $\mathcal{U}_{Y|X} = C_{YX}C_{XX}^{-1}$. While this assumption is satisfied for finite domains $\mathcal{X}$ with a characteristic kernel k, it does not necessarily hold when $\mathcal{X}$ is a continuous domain [Fukumizu et al., 2004], which is the case for many classification problems. In this case, $C_{YX}C_{XX}^{-1}$ becomes only an approximation to $\mathcal{U}_{Y|X}$, and we instead regularize the inversion and use $\mathcal{U}_{Y|X} = C_{YX}(C_{XX} + \lambda I)^{-1}$, which also serves to avoid overfitting [Song et al., 2013]. CMEs are useful for probabilistic inference as conditional expectations of a function $g \in \mathcal{H}_l$ can be expressed as inner products, $\mathbb{E}[g(Y)|X = x] = \langle \mu_{Y|X=x}, g \rangle$, provided that $\mathbb{E}[g(Y)|X = \cdot] \in \mathcal{H}_k$ [Song et al., 2009, Theorem 4].

Furthermore, as both C_{YX} and C_{XX} are defined in terms of expectations of kernels, we can replace them with their empirical estimates to derive a nonparametric estimate for $\mathcal{U}_{Y|X}$ based on finite collection of observations $\{x_i, y_i\} \in \mathcal{X} \times \mathcal{Y}$, $i \in \mathbb{N}_n := \{1, \dots, n\}$,

$$\hat{\mathcal{U}}_{Y|X} = \Psi(K + n\lambda I)^{-1}\Phi^T, \qquad (1)$$

where $K_{ij} := k(x_i, x_j)$, $\Phi := \begin{bmatrix}\phi(x_1) & \dots & \phi(x_n)\end{bmatrix}$, $\Psi := \begin{bmatrix}\psi(y_1) & \dots & \psi(y_n)\end{bmatrix}$, $\phi(x) := k(x, \cdot)$, and $\psi(y) := l(y, \cdot)$ [Song et al., 2013]. The empirical conditional embedding defined by $\hat{\mu}_{Y|X=x} := \hat{\mathcal{U}}_{Y|X}k(x, \cdot)$ then stochastically converges to the conditional embedding $\mu_{Y|X=x}$ in the RKHS norm at a rate of $O_p((n\lambda)^{-\frac{1}{2}} + \lambda^{\frac{1}{2}})$, under the assumption that $k(x, \cdot) \in \text{image}(C_{XX})$ [Song et al., 2009, Theorem 6]. This allows us to approximate the conditional expectation with $\langle \hat{\mu}_{Y|X=x}, g \rangle$ instead,

$$\mathbb{E}[g(Y)|X = x] \approx \langle \hat{\mu}_{Y|X=x}, g \rangle = \mathbf{g}^T(K + n\lambda I)^{-1}\mathbf{k}(x), \qquad (2)$$

where $\mathbf{g} := \{g(y_i)\}_{i=1}^n$ and $\mathbf{k}(x) := \{k(x_i, x)\}_{i=1}^n$.

2.2 Hyperparameter Learning

Hyperparameter learning for CMEs is particularly difficult compared to marginal or joint embeddings, since the kernel $k = k_\theta$ with parameters $\theta \in \Theta$ is to be learned jointly with a regularization parameter $\lambda \in \Lambda = \mathbb{R}_+$. Lever et al. [2012] proposed to hold out a validation set $\{k(x_{t_j}, \cdot), l(y_{t_j}, \cdot)\}_{j=1}^J$ and minimize $\frac{1}{J}\sum_{j=1}^J \left\| l(y_{t_j}, \cdot) - \hat{\mathcal{U}}_{Y|X}k(x_{t_j}, \cdot)\right\|_{\mathcal{H}_l}^2$ where $\hat{\mathcal{U}}_{Y|X}$ is estimated from the remaining training set using (1). This could also be repeated over multiple folds for cross validation. Song et al. [2013] also uses this cross validation approach, but adds a regularization $\lambda\|\mathcal{U}\|_{HS}^2$ to the validation objective. This approach improves generalization of CMEs to unseen examples. However, they require the selection for the split and number of folds. Moreover, they are not optimized for any particular prediction task. By fitting a separate model for each fold during learning, they also incur a large computational cost of $O(Jn^3)$ for J folds, and becomes prohibitive with large datasets.

When cross validation is too expensive, the length scale parameter can be set by the median heuristic [Muandet et al., 2016] via $\ell = \text{median}_{i,j}(\|x_i - x_j\|_2)$ for many stationary kernels. However, they cannot be used to set hyperparameters other than length scales, such as λ. In the setting of two sample testing, Gretton et al. [2012] also notes that they can possibly lead to poor performance. In the context of CMEs, they also do not make use of label information. Flaxman et al. [2016] proposed a Bayesian learning framework for marginal embeddings via inducing points, though it is unclear how this can be extended to CMEs. Fukumizu et al. [2009] also investigated the choice of kernel bandwidth for stationary kernels in the setting of binary classification and two sample testing using the maximum mean discrepancy (MMD), but has yet to generalize to CMEs or multiclass settings.

2.3 Rademacher Complexity

Rademacher complexity [Bartlett and Mendelson, 2002] measures the expressiveness of a function class F by its ability to shatter, or fit, noise. They are data-dependent measures, and are thus particularly well suited to learning tasks where generalization is vital, since complexity penalties that are not data dependent cannot be universally effective [Kearns et al., 1997].

In many cases, classifiers are trained by minimizing some empirical loss over a class of models whose Rademacher complexity is bounded. In the context of multi-label learning, Yu et al. [2014] used trace norm regularization to bound the Rademacher complexity, achieving tight generalization bounds. Xu et al. [2016] extends the trace norm regularization approach by considering the local Rademacher complexity on a subset of the predictor class, where they instead minimize the tail sum of the predictor singular values. Local Rademacher complexity has also been employed for multiple kernel learning [Cortes et al., 2013, Kloft and Blanchard, 2011] to learn convex combinations of fixed kernels for SVMs. Similarly, Pontil and Maurer [2013] also used trace norm regularization to bound the Rademacher complexity and minimize the truncated hinge loss. Nevertheless, while Rademacher complexities have been employed to restrict the function class considered for training weight parameters, they have not been applied to learn kernel hyperparameters itself.

3 Multi-Categorical Conditional Mean Embeddings

In this section we present a particular type of CMEs that are suitable for prediction tasks with categorical targets. We show that for CMEs with categorical targets and arbitrary inputs, we can further infer conditional probabilities directly, and not just conditional expectations. As there can be more than two categories, we refer to these CMEs as multi-categorical conditional embeddings (MCEs) in this paper.

For categorical targets, the output label space is finite and discrete, taking values only in $\mathcal{Y} = \mathbb{N}_m := \{1, \dots, m\}$. Naturally, we choose the Kronecker delta kernel $\delta : \mathbb{N}_m \times \mathbb{N}_m \to \{0, 1\}$ as the output kernel l, where labels that are the same have unit similarity and labels that are different have no similarity. That is, for all pairs of labels $y_i, y_j \in \mathcal{Y}$, $\delta(y_i, y_j) = 1$ only if $y_i = y_j$ and is 0 otherwise. As δ is an integrally strictly positive definite kernel on $\mathbb{N}_m$, it is therefore characteristic [Sriperumbudur et al., 2010, Theorem 7]. Therefore, by definition [Fukumizu et al., 2004], δ uniquely defines a RKHS $\mathcal{H}_\delta = \overline{\mathrm{span}\{\delta(y, \cdot) : y \in \mathcal{Y}\}}$, which is the closure of the span of its kernel induced features [Xu and Zhang, 2009]. For $\mathcal{Y} = \mathbb{N}_m$, this means that any $g : \mathbb{N}_m \to \mathbb{R}$ that is bounded on its discrete domain $\mathbb{N}_m$ is in the RKHS of δ, because we can always write $g = \sum_{y=1}^{m} g(y)\delta(y, \cdot) \in \mathrm{span}\{\delta(y, \cdot) : y \in \mathcal{Y}\}$. In particular, indicator functions on $\mathbb{N}_m$ are in $\mathcal{H}_\delta$, since $\mathbb{1}_c(y) := \mathbb{1}_{\{c\}}(y) = \delta(c, y)$. That is, indicator functions $\mathbb{1}_c = \delta(c, \cdot)$, $c \in \mathbb{N}_m$, are simply the canonical features of $\mathcal{H}_\delta$. Such properties do not necessarily hold for continuous target domains in general. For discrete target domains, this convenient property enables consistent estimations of decision probabilities.

Let $p_c(x) := \mathbb{P}[Y = c | X = x]$ be the *decision probability function* for class $c \in \mathbb{N}_m$, which is the probability of the class label Y being c when the example X is x. Note that there are no restrictions on the input domain $\mathcal{X}$ as long as a kernel k can be defined on it. For example, x could be a continuous variable in $\mathcal{X} = \mathbb{R}^d$. We begin by writing this probability as an expectation of indicator functions,

$$p_c(x) := \mathbb{P}[Y = c | X = x] = \mathbb{E}[\mathbb{1}_c(Y) | X = x]. \quad (3)$$

With $\mathbb{1}_c \in \mathcal{H}_\delta$, we let $g = \mathbb{1}_c$ in (2) and $\mathbf{1}_c := \{\mathbb{1}_c(y_i)\}_{i=1}^n$ to estimate the right hand side of (3) by

$$\hat{p}_c(x) = f_c(x) := \mathbf{1}_c^T (K + n\lambda I)^{-1} \mathbf{k}(x). \quad (4)$$

Let $\mathbf{Y} := \begin{bmatrix} \mathbf{1}_1 & \mathbf{1}_2 & \cdots & \mathbf{1}_m \end{bmatrix} \in \{0, 1\}^{n \times m}$ be the one hot encoded labels of $\{y_i\}_{i=1}^n$. The vector of empirical decision probabilities over the classes $c \in \mathbb{N}_m$ is then

$$\hat{\mathbf{p}}(x) = \mathbf{f}(x) := \mathbf{Y}^T (K + n\lambda I)^{-1} \mathbf{k}(x) \in \mathbb{R}^m. \quad (5)$$

Since $\mathcal{U} = \hat{\mathcal{U}}_{Y|X}$ (1) is the solution to a regularized least squares problem in the RKHS from $k(x, \cdot) \in \mathcal{H}_k$ to $l(y, \cdot) \in \mathcal{H}_l$ [Lever et al., 2012], CMEs can be seen as kernel ridged regressors (KRRs) in the RKHS. In this case, because $\mathcal{Y} = \mathbb{N}_m$ is discrete, $\mathcal{H}_\delta$ can be identified with $\mathbb{R}^m$. As a result, for discrete $\mathcal{Y} = \mathbb{N}_m$ the rows of the MCE can also be seen as m KRRs [Friedman et al., 2001] on binary $\{0, 1\}$-targets, where they all share the same input kernel. We now verify that the empirical decision probabilities (4) do converge to the population decision probability.

Theorem 3.1. *Assuming that $k(x, \cdot)$ is in the image of C_{XX}, the empirical decision probability function $\hat{p}_c : \mathcal{X} \to \mathbb{R}$ (4) converges uniformly to the true decision probability $p_c : \mathcal{X} \to [0, 1]$ (3) at a stochastic rate of at least $O_p((n\lambda)^{-\frac{1}{2}} + \lambda^{\frac{1}{2}})$ for all $c \in \mathcal{Y} = \mathbb{N}_m$. See Theorem A.2 of Appendix A for proof.*

In particular, the assumption $k(x, \cdot) \in \mathrm{image}(C_{XX})$ is on the input kernel k, not the output kernel l, which is a Kronecker delta $l = \delta$ for MCEs. In practice, this assumption can be relaxed through introducing the regularization parameter λ as in (1) [Song et al., 2009].

Note that for finite n the probability estimates (4) may not necessarily lie in the range $[0, 1]$. Although decision probability estimates (4) do not necessarily form a normalized distribution for finite n, theorem 3.1 guarantees that they approach one with increasing sample size. When normalized distributions are required, clip-normalized estimates can be used (4),

$$\tilde{p}_c(x) := \frac{\max\{\hat{p}_c(x), 0\}}{\sum_{j=1}^{m} \max\{\hat{p}_j(x), 0\}}. \tag{6}$$

This does not change the resulting prediction, since $\hat{y}(x) = \operatorname{argmax}_{c \in \mathbb{N}_m} \hat{p}_c(x) = \operatorname{argmax}_{c \in \mathbb{N}_m} \tilde{p}_c(x)$. Theorem 3.1 also implies that eventually the effect of clip-normalization vanishes, where $\tilde{p}_c(x)$ approaches to both $\hat{p}_c(x)$ and $p_c(x)$ with increasing sample sizes.

This also allows MCEs to be naturally applied to perform probabilistic classification in multiclass settings with categorical targets. In contrast, support vector classifiers (SVCs) do not naturally produce probabilities, while Gaussian process classifiers (GPCs) require posterior approximations. Furthermore, multiclass extensions to SVCs and GPCs often employ the one versus all (OVA) or one versus one (OVO) scheme [Aly, 2005], resulting in multiple separate binary classifiers. Instead, fitting a single MCE is sufficient for producing consistent multiclass probabilistic estimates.

Similar to RLSC, MCEs are solutions to a regularized least squares problem in a RKHS [Lever et al., 2012], resulting in a similar system of linear equations. RLSCs primarily differ in the way they handle the labels, in which binary labels $\{-1, 1\}$ appear directly in the squared loss instead of its kernel feature $\delta(y_i, \cdot)$ or, equivalently, its one hot encoded form $\mathbf{y}_i$. Consequently, multiclass extensions for RLSC either require using the OVA scheme [Rifkin et al., 2003], or using the summed losses across all binarized tasks for the overall least squares problem [Pahikkala et al., 2012], producing separate classifiers for each class. As a result, multiclass RLSC does not produce consistent estimates of class probabilities as in theorem 3.1 for MCEs.

4 Hyperparameter Learning with Rademacher Complexity Bounds

In this section we derive learning theoretic bounds that motivates our proposed hyperparameter learning algorithm, and discuss how it can be extended in various ways to enhance scalability and performance. From here onwards, we denote θ as the kernel hyperparameters of the kernel $k = k_\theta$.

We begin by defining a loss function as a measure for performance. For decision functions of the form $\mathbf{f} : \mathcal{X} \to \mathcal{A} = \mathbb{R}^m$ whose entries are probability estimates, we employ a modified cross entropy loss,

$$\mathcal{L}_\epsilon(y, \mathbf{f}(x)) := -\log{[\mathbf{y}^T \mathbf{f}(x)]}_\epsilon^1 = -\log{[f_y(x)]}_\epsilon^1, \tag{7}$$

to express risk, where we use the notation $[\ \cdot\ ]_\epsilon^1 := \min\{\max\{\ \cdot\ , \epsilon\}, 1\}$ for $\epsilon \in (0, 1)$ where usually $\epsilon << 1$. Note that we employ the loss on the original probability estimates (5), not the clip-normalized version (6). We employ this loss in virtue of theorem 3.1, where we expect $\mathbf{f}(x)$ (5) to be close to the population decision probability. In contrast, direct outputs from SVCs, GPCs, or RLSCs are not consistent probability estimates and cannot take advantage of (7) easily.

However, simply minimizing the empirical loss $\frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_\epsilon(y_i, \mathbf{f}_{\theta,\lambda}(x_i))$ over the hyperparameters (θ, λ) could lead to an overfitted model. We therefore employ Rademacher complexity bounds to control the model complexity of MCEs.

Let Θ and Λ be a space of kernel hyperparameters and regularization parameters respectively. We define the class of MCEs over these hyperparameter spaces by

$$F_n(\Theta, \Lambda) := \{\mathbf{f}_{\theta,\lambda}(x) : \theta \in \Theta, \lambda \in \Lambda\}. \tag{8}$$

We denote $W_{\theta,\lambda}^T \equiv \hat{\mathcal{U}}_{Y|X}^{(\theta,\lambda)}$ and $\|W_{\theta,\lambda}\|_{\mathrm{tr}} = \|\hat{\mathcal{U}}_{Y|X}^{(\theta,\lambda)}\|_{HS}$ to reflect the dependence on (θ, λ) and for notational simplicity. We first restrict the space of hyperparameters by the norms of $W_{\theta,\lambda}$ and $k_\theta(x, x)$ to obtain an upper bound to the Rademacher complexity of $F_n(\Theta, \Lambda)$.

Theorem 4.1. *Assume that $\|W_{\theta,\lambda}\|_{\mathrm{tr}} \leq \rho$ is bounded for all $\theta \in \Theta, \lambda \in \Lambda$. Further suppose that the canonical feature map $\|\phi_\theta(x)\|_{\mathcal{H}_{k_\theta}}^2 = k_\theta(x, x) \leq \alpha^2$, $\alpha > 0$, is bounded in RKHS norm for all $x \in \mathcal{X}, \theta \in \Theta$. For any set of training observations $\{x_i, y_i\}_{i=1}^n$, the Rademacher complexity of the class of MCEs $F_n(\Theta, \Lambda)$ (8) defined over $\theta \in \Theta, \lambda \in \Lambda$ is bounded by*

$$\mathcal{R}_n(F_n(\Theta, \Lambda)) \leq 2\alpha\rho. \tag{9}$$

See Theorem B.1 of Appendix B for proof.

Bartlett and Mendelson [2002] showed that the expected risk can be bounded with high probability using the empirical risk and the Rademacher complexity of the loss composed with the function class. For a Lipchitz loss, Ledoux and Talagrand [2013] further showed that the latter quantity can be bounded using the Rademacher complexity of the function class itself. We use these two results to arrive at the following probabilistic upper bound to our expected loss. We refer the reader to Appendix B for detailed discussion.

Theorem 4.2. *Assume the same assumptions as theorem 4.1. For any integer $n \in \mathbb{N}_+$, any $\epsilon \in (0, e^{-1})$, and any set of training observations $\{x_i, y_i\}_{i=1}^n$, with probability of at least $1 - \beta$ over iid samples $\{X_i, Y_i\}_{i=1}^n$ of length n from $\mathbb{P}_{XY}$, every $f \in F_n(\Theta, \Lambda)$ satisfies*

$$\mathbb{E}[\mathcal{L}_{e^{-1}}(Y, f(X))] \\ \leq \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_{\epsilon}(Y_i, f(X_i)) + 4e\,\alpha\rho + \sqrt{\frac{8}{n}\log\frac{2}{\beta}}. \quad (10)$$

See Theorem B.6 of Appendix B for proof.

However, for hyperparameter learning, we would require a risk bound for specific choice of hyperparameters, not just for a set of hyperparameters. For some $\tilde{\theta} \in \Theta$ and $\tilde{\lambda} \in \Lambda$, we construct a subset of hyperparameters $\Xi(\tilde{\theta}, \tilde{\lambda}) \subseteq \Theta \times \Lambda$ defined by $\Xi(\tilde{\theta}, \tilde{\lambda}) := \{(\theta, \lambda) \in \Theta \times \Lambda : \|W_{\theta,\lambda}\|_{\mathrm{tr}} \leq \|W_{\tilde{\theta},\tilde{\lambda}}\|_{\mathrm{tr}},\ \sup_{x\in\mathcal{X}} k_\theta(x,x) \leq \alpha^2(\tilde{\theta}) := \sup_{x\in\mathcal{X}} k_{\tilde{\theta}}(x,x)\}$. Clearly, this subset is non-empty, since $(\tilde{\theta}, \tilde{\lambda}) \in \Xi(\tilde{\theta}, \tilde{\lambda})$ is itself an element of this subset. Thus, we can assert that $\|W_{\theta,\lambda}\|_{\mathrm{tr}} \leq \rho = \|W_{\tilde{\theta},\tilde{\lambda}}\|_{\mathrm{tr}}$ is bounded for all $(\theta, \lambda) \in \Xi(\tilde{\theta}, \tilde{\lambda})$, and that $\|\phi_\theta(x)\|^2_{\mathcal{H}_{k_\theta}} = k_\theta(x,x) \leq \alpha^2 = \sup_{x\in\mathcal{X}} k_{\tilde{\theta}}(x,x)$ is bounded for all $x \in \mathcal{X}, (\theta,\lambda) \in \Xi(\tilde{\theta}, \tilde{\lambda})$.

We can now choose $\tilde{\theta} \in \Theta$, $\tilde{\lambda} \in \Lambda$ and apply theorem 4.2 with the choice of $\rho = \|W_{\tilde{\theta},\tilde{\lambda}}\|_{\mathrm{tr}}$ and $\alpha^2 = \sup_{x\in\mathcal{X}} k_{\tilde{\theta}}(x,x)$ and by considering only the hyperparameters $(\theta, \lambda) \in \Xi(\tilde{\theta}, \tilde{\lambda})$. The probabilistic statement (10) then only holds for (θ, λ) within $\Xi(\tilde{\theta}, \tilde{\lambda})$. In particular, since $(\tilde{\theta}, \tilde{\lambda}) \in \Xi(\tilde{\theta}, \tilde{\lambda})$, it holds for $\theta = \tilde{\theta}$ and $\lambda = \tilde{\lambda}$. Applying this choice, the only variables that remain now are $(\tilde{\theta}, \tilde{\lambda})$. We then replace these symbols with a general (θ, λ) again to avoid cluttered notation. We now arrive at our final result.

Theorem 4.3. *For any integer $n \in \mathbb{N}_+$ and any set of training observations $\{x_i, y_i\}_{i=1}^n$ used to define $\mathbf{f}_{\theta,\lambda}$ (5), with probability $1-\beta$ over iid samples $\{X_i, Y_i\}_{i=1}^n$ of length n from $\mathbb{P}_{XY}$, every $\theta \in \Theta$, $\lambda \in \Lambda$, and $\epsilon \in (0, e^{-1})$ satisfies*

$$\mathbb{E}[\mathcal{L}_{e^{-1}}(Y, \mathbf{f}_{\theta,\lambda}(X))] \\ \leq \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_{\epsilon}(Y_i, \mathbf{f}_{\theta,\lambda}(X_i)) + 4e\,r(\theta,\lambda) + \sqrt{\frac{8}{n}\log\frac{2}{\beta}}, \quad (11)$$

where $r(\theta, \lambda) := \sqrt{\mathrm{trace}(V_{\theta,\lambda}^T K_\theta V_{\theta,\lambda}) \sup_{x\in\mathcal{X}} k_\theta(x,x)}$ and $V_{\theta,\lambda} := (K_\theta + n\lambda I)^{-1}\mathbf{Y}$. See Theorem B.8 of Appendix B for proof.

In particular, $r(\theta, \lambda)$ is an upper bound to the Rademacher complexity of a relevant class of MCEs based on the hyperparameters $\Xi(\theta, \lambda)$. We call $r(\theta, \lambda)$ the Rademacher complexity bound (RCB) and use it to measure the model complexity of a MCE with hyperparameters (θ, λ). Since the training set itself is a sample of length n drawn from $\mathbb{P}_{XY}$, the inequality (11) holds with probability $1-\beta$ when the random variables (X_i, Y_i) are realized as the training observations (x_i, y_i).

Algorithm 1 MCE Hyperparameter Learning with Batch Stochastic Gradient Updates

```
1:  Input: kernel family $k_\theta : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, dataset $\{x_i, y_i\}_{i=1}^n$, initial kernel parameters $\theta_0$, initial regularization parameter $\lambda_0$, learning rate $\eta$, cross entropy loss threshold $\epsilon$, batch size $n_b$
2:  $\theta \leftarrow \theta_0$, $\lambda \leftarrow \lambda_0$
3:  repeat
4:      Sample the next batch $\mathcal{I}_b \subseteq \mathbb{N}_n$ s.t. $|\mathcal{I}_b| = n_b$
5:      $Y \leftarrow \{\delta(y_i, c) : i \in \mathcal{I}_b, c \in \mathbb{N}_m\}$    $\in \{0,1\}^{n_b \times m}$
6:      $K_\theta \leftarrow \{k_\theta(x_i, x_j) : i \in \mathcal{I}_b, j \in \mathcal{I}_b\}$    $\in \mathbb{R}^{n_b \times n_b}$
7:      $L_{\theta,\lambda} \leftarrow \mathrm{cholesky}(K_\theta + n_b \lambda I_{n_b})$    $\in \mathbb{R}^{n_b \times n_b}$
8:      $V_{\theta,\lambda} \leftarrow L_{\theta,\lambda}^T \backslash (L_{\theta,\lambda} \backslash Y)$    $\in \mathbb{R}^{n_b \times m}$
9:      $P_{\theta,\lambda} \leftarrow K_\theta V_{\theta,\lambda}$    $\in \mathbb{R}^{n_b \times m}$
10:     $r(\theta,\lambda) \leftarrow \alpha(\theta)\sqrt{\mathrm{trace}(V_{\theta,\lambda}^T K_\theta V_{\theta,\lambda})}$
11:     $q(\theta,\lambda) \leftarrow \frac{1}{n_b}\sum_{i=1}^{n_b} \mathcal{L}_\epsilon((Y)_i, (P_{\theta,\lambda})_i) + 4e\,r(\theta,\lambda)$
12:     $(\theta,\lambda) \leftarrow \mathrm{GradientBasedUpdate}(q, \theta, \lambda; \eta)$
13: until maximum iterations reached
14: Output: kernel parameters $\theta$, regularization $\lambda$
```

Motivated by this, we employ this upper bound as the learning objective for hyperparameter learning,

$$q(\theta, \lambda) := \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_\epsilon(y_i, \mathbf{f}_{\theta,\lambda}(x_i)) + 4e\,r(\theta, \lambda). \quad (12)$$

In particular, the first term is an empirical risk that measures data fit, and the second term is the RCB that measures model complexity.

4.1 Extensions

Batch Stochastic Gradient Update Since theorem 4.3 holds for any $n \in \mathbb{N}_+$ and any set of data $\{x_i, y_i\}_{i=1}^n$ from $\mathbb{P}_{XY}$, the bound (11) also holds with high probability for a batch subset of the training data. However, the batch size cannot be too small, in order to keep the constant $\sqrt{8\log(2/\beta)/n}$ relatively small. We therefore propose to use only a random batch subset of the data to perform each gradient update. This enables scalable hyperparameter learning through batch stochastic gradient updates, where each gradient update stochastically improves a different probabilistic upper bound of the generalization risk. Note that without this bound, it is not straightforward to simply apply stochastic gradient updates to optimize q, since r depends on the dataset but is not written in terms of a summation over the data. We present this scalable hyperparameter learning approach via batch stochastic gradient updates in algorithm 1, reducing the time complexity from $O(n^3)$ to $O(n_b^3)$, where n_b is the batch size. The Cholesky decomposition for the full training set requires $O(n^3)$ time and is necessary only for inference, instead of once every learning iteration. It can

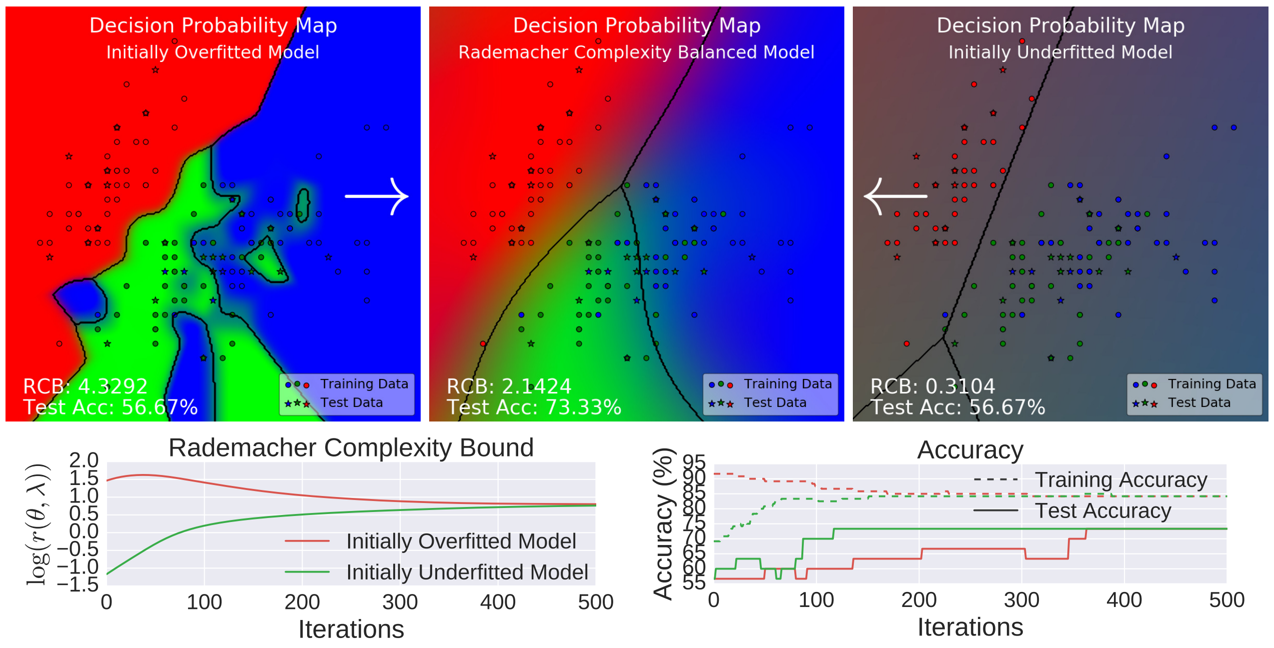

Figure 1: Rademacher complexity balanced learning of hyperparameters for an isotropic Gaussian MCE, using the first two attributes of the iris dataset.

be further avoided by using random Fourier features [Rahimi and Recht, 2008] or kernel herding [Chen et al., 2010] to approximate the already learned MCE. All further inference takes $O(n^2)$ time, or potentially less with approximation, using back substitution.

Conditional Embedding Network Our learning algorithm does not restrict the way the kernel k_θ is constructed from its parameters $\theta \in \Theta$. One particularly useful type of MCEs are those where the input kernel $k_\theta(x, x') = \langle \varphi_\theta(x), \varphi_\theta(x') \rangle$ is constructed from neural networks $\varphi_\theta : \mathcal{X} \to \mathbb{R}^p$ explicitly. We refer to MCEs constructed this way as conditional embedding networks (CENs). In these cases, the weights and biases of the neural network become the kernel parameters θ of the CENs. We can therefore network weights and biases jointly under (12). CENs can also scale easily, since the $n \times n$ Cholesky decomposition required for full gradient updates in algorithm 1 can be transformed into a $p \times p$ decomposition by the Woodbury matrix inversion identity [Higham, 2002], reducing the time complexity to $O(p^3 + np^2)$. This allows scalable learning for $n >> p$ even without using batch gradient updates. For inference, standard map reduce methods can be used. We direct the reader to appendix C for detailed discussion on the various MCE architectures and their implementation as compared to algorithm 1.

Batch Validation While we simply instantiated (X_i, Y_i) to be the training observations in theorem 4.3 to obtain (12), this does not have to be the case for batch updates. Instead, in each learning iteration, we could further split the batch into two sub-batches – one for training and one for validation. The training batch is used to form the MCE $\mathbf{f}_{\theta,\lambda}$ and RCB $r(\theta, \lambda)$, while we evaluate the empirical risk on the validation batch,

$$q^V(\theta, \lambda) := \frac{1}{n_V} \sum_{i=1}^{n_V} \mathcal{L}_\epsilon(y_i^V, \mathbf{f}_{\theta,\lambda}^T(x_i^V)) + \tau\, r^T(\theta, \lambda), \tag{13}$$

where (T) and (V) denotes training and validation. Note that in contrast to standard cross validation, not all data is required for each update due to the presence of the RCB. Furthermore, although the multiplier on the RCB is $4e$, experiments show that generalization performance can improve if we uses a smaller multiplier $\tau < 4e$. In practice, these two extensions work well together. Intuitively, by introducing a validation batch to measure empirical data fit, a smaller weight on the complexity penalty is required.

5 Experiments

Toy Example The first two of four total attributes of the iris dataset [Fisher, 1936] are known to have class labels that are non-separable by any means, in that the same example $x \in \mathbb{R}^2$ may be assigned different output labels $y \in \mathbb{N}_3 := \{1, 2, 3\}$. In these difficult scenarios, the notion of model complexity is extremely important, and the success of a learning algorithm greatly depends on how it balances training performance and model

Table 1: Test accuracy (%) on UCI datasets

Dataset	GMCE	GMCE-SGD	CEN-1	CEN-2	ERM	CV	MED	Others
`banknote`	$\mathbf{99.9 \pm 0.2}$	98.8 ± 0.9	99.5 ± 1.0	99.4 ± 0.9	$\mathbf{99.9 \pm 0.2}$	$\mathbf{99.9 \pm 0.2}$	92.0 ± 4.3	99.78[a]
`ecoli`	$\mathbf{87.5 \pm 4.4}$	84.5 ± 5.0	$\mathbf{87.5 \pm 3.2}$	86.3 ± 6.0	72.1 ± 20.5	73.8 ± 23.8	42.1 ± 47.7	81.1[b]
`robot`	$\mathbf{96.7 \pm 0.9}$	95.5 ± 0.9	82.3 ± 7.1	94.5 ± 0.8	91.0 ± 3.7	90.9 ± 3.4	81.1 ± 6.2	97.59[c]
`segment`	$\mathbf{98.4 \pm 0.8}$	96.1 ± 1.5	94.6 ± 1.6	96.7 ± 1.1	98.1 ± 1.1	98.3 ± 1.3	27.3 ± 26.4	96.83[d]
`wine`	$\mathbf{97.2 \pm 3.7}$	93.3 ± 6.0	96.1 ± 5.0	97.2 ± 5.1	93.9 ± 5.2	93.3 ± 7.4	93.3 ± 7.8	100[e]
`yeast`	52.5 ± 2.1	$\mathbf{60.3 \pm 4.4}$	55.8 ± 5.0	59.6 ± 4.0	45.9 ± 6.4	58.0 ± 5.8	31.2 ± 14.1	55.0[b]

complexity to avoid both underfitting and overfitting.

Figure 1 demonstrates algorithm 1 with full gradient updates ($n_b = n$) to learn hyperparameters of the MCE on the two attribute iris dataset. The kernel used is isotropic Gaussian with diagonal length scales $\Sigma = \ell^2 I_2$ and sensitivity $\alpha = \sigma_f$, so that the hyperparameters are $\theta = (\alpha, \ell)$ and λ. We evaluate the performance of the learning algorithm on a withheld test set using 20% of the available 150 data samples. Attributes are scaled into the unit range $[0, 1]$ and decision probability maps are plotted for the region $[-0.5, 1.05]^2$, where the red, green, and blue color channels represent the clip-normalized decision probability (6) for classes $c = 1, 2, 3$. We begin from two initial sets of hyperparameters, one originally overfitting and another underfitting the training data. Initially, both models perform sub-optimally with a test accuracy of 56.67%. We see that the RCB $r(\theta, \lambda)$ appropriately measures the amount of overfitting with high (resp. low) values for the overfitted (resp. underfitted) model. We then learn hyperparameters with algorithm 1 for 500 iterations from both initializations at rate $\eta = 0.01$, where both models converges to a balanced model with a moderate RCB and an improved test accuracy of 73.33%. In particular, the initially overfitted model learns a simpler model at the expense of lower training performance, emphasizing the benefits of complexity based regularization, without which the learning would only maximize training performance at the cost of further overfitting. Meanwhile, the initially underfitted model learns to increase complexity to improve the sub-optimal performance on the training set.

UCI Datasets We demonstrate the average performance of learning anisotropic Gaussian kernels and kernels constructed from neural networks on standard UCI datasets [Bache and Lichman, 2013], summarized in table 1. The former has a shallow but wide model architecture, while the latter has a deeper but narrower model architecture. The Gaussian kernel is learned with both full (GMCE) and batch stochastic gradient updates (GMCE-SGD) using a tenth ($n_b \approx \frac{n}{10}$) of the training set each training iteration, with sensitivity and length scales initialized to 1. For CENs, we randomly select two simple fully connected architectures with 16-32-8 (CEN-1) and 96-32 (CEN-2) hidden units respectively, and learn the conditional embedding without dropout under ReLU activation. Biases and standard deviations of zero mean truncated normal distributed weights are initialized to 0.1, and are to be learned with full gradient updates. For all experiments, λ is initialized to 1 and is learned jointly with the kernel. Optimization is performed with the Adam optimizer [Kingma and Ba, 2016] in TensorFlow [Abadi et al., 2016] with a rate of $\eta = 0.1$ and $\epsilon = 10^{-15}$ under the learning objective $q(\theta, \lambda)$ (12). Learning is run for 1000 epochs to allow direct comparison. All attributes are scaled to the unit range. Each model is trained on 9 out of 10 folds and tested on the remaining fold, which are shuffled over all 10 combinations to obtain the test accuracy average and deviation. We compare our results to MCEs whose hyperparameters are tuned by ERM (without the RCB term in (12)), cross validation (CV), and the median heuristic (MED), as well as to other approaches using neural networks [Freire et al., 2009, Kaya et al., 2016, a; c], probabilistic binary trees [Horton and Nakai, 1996, b], decision trees [Zhou et al., 2004, d], and regularised discriminant analysis [Aeberhard et al., 1992, e]. Table 1 shows that our learning algorithm outperforms other hyperparameter tuning algorithms, and performs similarly to competing methods. Our method achieves this without any special tuning or heuristics, but by simply placing a conditional embedding on training data and applying a complexity bound based learning algorithm. The stochastic gradient approach for Gaussian kernels performs similarly to the full gradient approach, supporting theorem 4.3 for $n = n_b$. For CENs, we did not attempt to choose an optimal architecture for each dataset. The learning algorithm is designed to train the same simple network for different datasets using 1000 epochs to generate comparable performance.

Learning pixel relevance We apply algorithm 1 to learn length scales of anisotropic Gaussian, or ARD, kernels on pixels of the MNIST digits dataset [LeCun et al., 1998]. In the top left plot of fig. 2, we train on datasets of varying sizes, from 50 to 5000 images, and show the accuracy on the standard test set of 10000

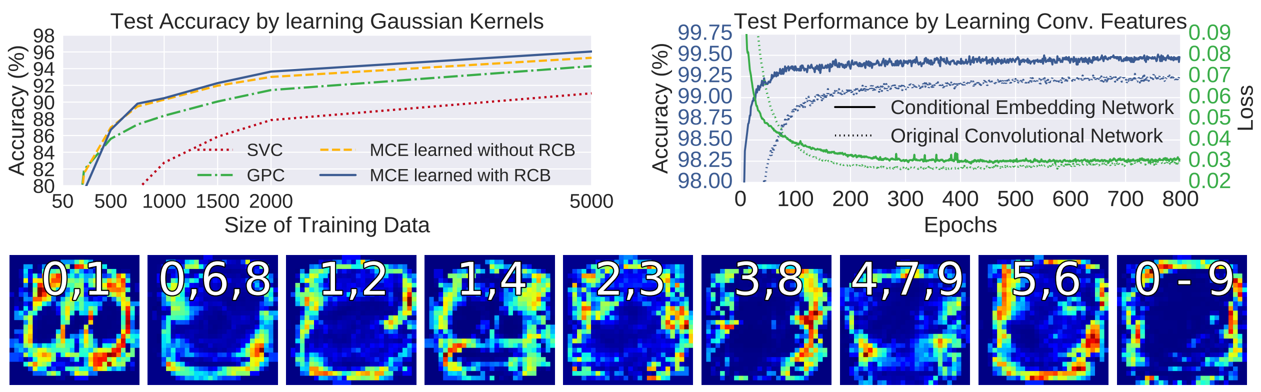

Figure 2: Top: Test accuracy by learning Gaussian kernels (left) and deep convolutional features (right). Bottom: Learned pixel length scales under anistropic Gaussian, or ARD, kernels.

images. All hyperparameters are initialized to 1 before learning. We train both SVCs and GPCs under the OVA scheme, and use a Laplace approximation for the GPC posterior. In all cases MCEs outperform SVCs as it cannot learn hyperparameters without expensive cross validation. MCEs also outperform GPCs as more data becomes available. Under the OVA scheme, the GPC approach learns a set of kernel hyperparameters for each class, while our approach learns a consistent set of hyperparameters for all classes. Consequently, for 5000 data points, the computational time required for hyperparameter learning of GPCs is on the order of days even for isotropic Gaussian kernels, while algorithm 1 is on the order of hours for anistropic Gaussian kernels even without batch updates. We also compare hyperparameter learning with and without the RCB. For small n below 750 samples, the latter outperforms the former (e.g. 86.69% and 86.96% for $n = 500$), while for large n the former outperforms the latter (e.g. 96.05% and 95.3% for $n = 5000$). This verifies that complexity based regularization becomes especially important as data size grows, when overfitting starts to decrease generalization performance. The images at the bottom of fig. 2 show the pixel length scales learned through batch stochastic gradient updates ($n_b = 1200$) over all available training images the groups of digits shown, demonstrating the most discriminative regions.

Learning convolutional layers We now apply algorithm 1 to train a CEN with convolutional layers on MNIST. We employ an example architecture from the TensorFlow tutorial on deep MNIST classification [Abadi et al., 2016]. This ReLU activated convolutional neural network (CNN) uses two convolutional layers, each with max pooling, followed by a fully connected layer with a drop out probability of 0.5. The original CNN then employs a final softmax regressor on the last hidden layer for classification. The CEN instead employs a linear kernel on the last hidden layer to construct the conditional embedding. We then train both networks from the same initialization using batch updates of $n_b = 6000$ images for 800 epochs, with learning rate $\eta = 0.01$. All biases and weight standard deviations are initialized to 0.1. The network weights and biases of the CEN are learned jointly with the regularization parameter, initialized to $\lambda = 10$, under our learning objective (12), while the original CNN is trained under its usual cross entropy loss. The fully connected layer is trained with a drop out probability of 0.5 for both cases to allow direct comparison. The top right plot in fig. 2 shows that CENs learn at a much faster rate, maintaining a higher test accuracy at all epochs. After 800 epochs, CEN reaches a test accuracy of 99.48%, compared to 99.26% from the original CNN. This demonstrates that our learning algorithm can perform end-to-end learning with convolutional layers from scratch, by simply replacing the softmax layer with a MCE. The resulting CEN can outperform the original CNN in both convergence rate and accuracy.

6 Conclusion and Future Work

We developed a scalable hyperparameter learning framework for CMEs with categorical targets based on Rademacher complexity bounds. These bounds reveal a novel data-dependent quantity $r(\theta, \lambda)$ which reflect its model complexity, which we use as an regularization term in addition to the empirical loss for hyperparameter learning. Similar to a regularized least squares problem, it remains to be established what type of prior on the hyperparameters, if any, could correspond to such a regularizer, resulting in a Bayesian interpretation of our approach. We also envision that such a quantity could potentially be generalized to CMEs with arbitrary targets, which would enable hyperparameter learning for general conditional mean embeddings in a way that is optimized for the prediction task.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). TensorFlow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA*.

Aeberhard, S., Coomans, D., and De Vel, O. (1992). Comparison of classifiers in high dimensional settings. *Dept. Math. Statist., James Cook Univ., North Queensland, Australia, Tech. Rep*, (92-02).

Aly, M. (2005). Survey on multiclass classification methods. *Neural Netw*, pages 1–9.

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

Bartlett, P. L. and Mendelson, S. (2002). Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482.

Chen, Y., Welling, M., and Smola, A. (2010). Super-samples from kernel herding. In *The Twenty-Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-10)*, pages 109–116. AUAI Press.

Cortes, C., Kloft, M., and Mohri, M. (2013). Learning kernels using local Rademacher complexity. In *Advances in neural information processing systems*, pages 2760–2768.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188.

Flaxman, S., Sejdinovic, D., Cunningham, J. P., and Filippi, S. (2016). Bayesian learning of kernel embeddings. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pages 182–191. AUAI Press.

Freire, A. L., Barreto, G. A., Veloso, M., and Varela, A. T. (2009). Short-term memory mechanisms in neural network learning of robot navigation tasks: A case study. In *Robotics Symposium (LARS), 2009 6th Latin American*, pages 1–6. IEEE.

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics New York.

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

Fukumizu, K., Gretton, A., Lanckriet, G. R., Schölkopf, B., and Sriperumbudur, B. K. (2009). Kernel choice and classifiability for rkhs embeddings of probability distributions. In *Advances in neural information processing systems*, pages 1750–1758.

Fukumizu, K., Song, L., and Gretton, A. (2013). Kernel Bayes' rule: Bayesian inference with positive definite kernels. *Journal of Machine Learning Research*, 14(1):3753–3783.

Gretton, A., Borgwardt, K. M., Rasch, M., Schölkopf, B., and Smola, A. J. (2007). A kernel method for the two-sample-problem. In *Advances in neural information processing systems*, pages 513–520.

Gretton, A., Sejdinovic, D., Strathmann, H., Balakrishnan, S., Pontil, M., Fukumizu, K., and Sriperumbudur, B. K. (2012). Optimal kernel choice for large-scale two-sample tests. In *Advances in neural information processing systems*, pages 1205–1213.

Higham, N. J. (2002). *Accuracy and stability of numerical algorithms*. SIAM.

Horton, P. and Nakai, K. (1996). A probabilistic classification system for predicting the cellular localization sites of proteins. In *Ismb*, volume 4, pages 109–115.

Jitkrittum, W., Szabó, Z., Chwialkowski, K. P., and Gretton, A. (2016). Interpretable Distribution Features with Maximum Testing Power. In *Advances In Neural Information Processing Systems*, pages 181–189.

Kanagawa, M. and Fukumizu, K. (2014). Recovering Distributions from Gaussian RKHS Embeddings. In *AISTATS*, pages 457–465.

Kanagawa, M., Nishiyama, Y., Gretton, A., and Fukumizu, K. (2016). Filtering with state-observation examples via kernel monte carlo filter. *Neural computation*, 28(2):382–444.

Kaya, E., Yasar, A., and Saritas, I. (2016). Banknote Classification Using Artificial Neural Network Approach. *International Journal of Intelligent Systems and Applications in Engineering*, 4(1):16–19.

Kearns, M., Mansour, Y., Ng, A. Y., and Ron, D. (1997). An experimental and theoretical comparison of model selection methods. *Machine Learning*, 27(1):7–50.

Kingma, D. and Ba, J. (2016). Adam: A method for stochastic optimization. *The International Conference on Learning Representations (ICLR)*.

Kloft, M. and Blanchard, G. (2011). The local Rademacher complexity of lp-norm multiple kernel learning. In *Advances in Neural Information Processing Systems*, pages 2438–2446.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Ledoux, M. and Talagrand, M. (2013). *Probability in Banach Spaces: isoperimetry and processes*. Springer Science & Business Media.

Lever, G., Baldassarre, L., Patterson, S., Gretton, A., Pontil, M., and Grünewälder, S. (2012). Conditional mean embeddings as regressors. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1823–1830.

Muandet, K., Balduzzi, D., and Schölkopf, B. (2013). Domain Generalization via Invariant Feature Representation. In *ICML (1)*, pages 10–18.

Muandet, K., Fukumizu, K., Sriperumbudur, B., and Schölkopf, B. (2016). Kernel Mean Embedding of Distributions: A Review and Beyonds. *arXiv preprint arXiv:1605.09522*.

Pahikkala, T., Airola, A., Gieseke, F., and Kramer, O. (2012). Unsupervised multi-class regularized least-squares classification. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 585–594. IEEE.

Pontil, M. and Maurer, A. (2013). Excess risk bounds for multitask learning with trace norm regularization. In *Conference on Learning Theory*, pages 55–76.

Rahimi, A. and Recht, B. (2008). Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian processes for machine learning*. The MIT Press.

Rifkin, R., Yeo, G., Poggio, T., et al. (2003). Regularized least-squares classification. *Nato Science Series Sub Series III Computer and Systems Sciences*, 190:131–154.

Schölkopf, B. and Smola, A. J. (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Song, L., Fukumizu, K., and Gretton, A. (2013). Kernel embeddings of conditional distributions: A unified kernel framework for nonparametric inference in graphical models. *IEEE Signal Processing Magazine*, 30(4):98–111.

Song, L., Huang, J., Smola, A., and Fukumizu, K. (2009). Hilbert space embeddings of conditional distributions with applications to dynamical systems. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 961–968. ACM.

Song, L., Zhang, X., Smola, A., Gretton, A., and Schölkopf, B. (2008). Tailoring density estimation via reproducing kernel moment matching. In *Proceedings of the 25th international conference on Machine learning*, pages 992–999. ACM.

Sriperumbudur, B. K., Gretton, A., Fukumizu, K., Schölkopf, B., and Lanckriet, G. R. (2010). Hilbert space embeddings and metrics on probability measures. *Journal of Machine Learning Research*, 11(Apr):1517–1561.

Xu, C., Liu, T., Tao, D., and Xu, C. (2016). Local Rademacher complexity for multi-label learning. *IEEE Transactions on Image Processing*, 25(3):1495–1507.

Xu, Y. and Zhang, H. (2009). Refinement of reproducing kernels. *Journal of Machine Learning Research*, 10(Jan):107–140.

Yu, H.-f., Jain, P., Kar, P., and Dhillon, I. (2014). Large-scale Multi-label Learning with Missing Labels. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 593–601.

Zhou, Z.-H., Wei, D., Li, G., and Dai, H. (2004). On the size of training set and the benefit from ensemble. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 298–307. Springer.