Hyperparameter Learning for Conditional Mean Embeddings with Rademacher Complexity Bounds

Kelvin Hsu, Richard Nock, Fabio Ramos

Research Summary

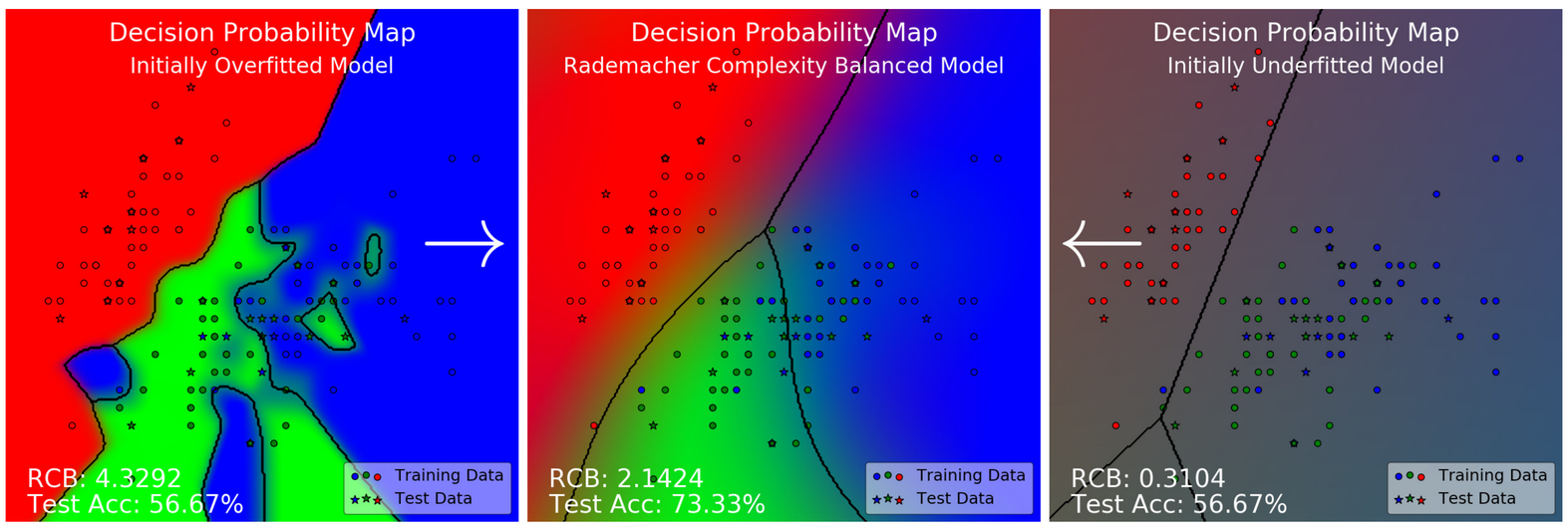
Background: Conditional mean embeddings (CMEs) are kernel models that nonparametrically encode expectations under conditional distributions, forming a flexible and powerful framework for probabilistic inference.

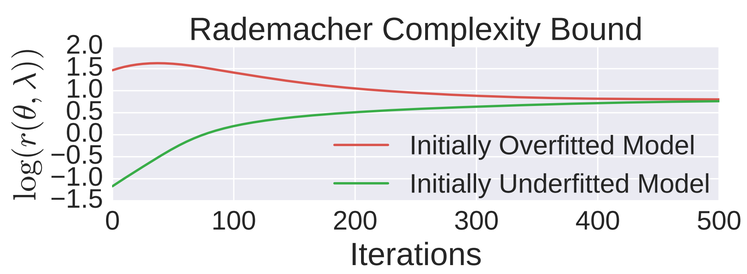
Problem: Their hyperparameters are notoriously difficult to tune or learn.

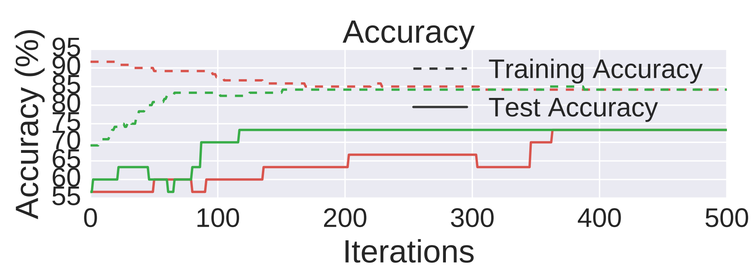
Question: Can we design a scalable hyperparameter learning algorithm for CMEs to ensure good generalization?

Contribution: We show that when CMEs are used to estimate multiclass probabilities, there are learning-theoretic bounds based on Rademacher complexities that result in a complexity-based hyperparameter learning algorithm which 1) balances data fit and model complexity, 2) amends to batch stochastic gradient updates, and 3) demonstrates capability to learn more flexible kernels such as those constructed from neural networks.

Toy Example: Non-Separable Iris







Setup: The data is non-separable *by any means* – the same $x \in \mathbb{R}^2$ may be assigned different labels $y \in \{1, 2, 3\}$. It is very easy for models to overfit by forcing a pattern or underfit by giving up.

Result: Our learning algorithm can drive the model from any initial state, overfitted (left) or underfitted (right), to a complexity balanced state where generalization accuracy is the highest.

Algorithm

Input: kernel family $k_\theta : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, dataset $\{x_i, y_i\}_{i=1}^n$, initial kernel hyperparameters θ_0, initial regularization hyperparameter λ_0, learning rate η, cross entropy loss threshold ϵ, batch size n_b

$\theta \leftarrow \theta_0$, $\lambda \leftarrow \lambda_0$

repeat

  Sample the next batch $\mathcal{I}_b \subseteq \mathbb{N}_n$, $|\mathcal{I}_b| = n_b$

  $Y \leftarrow \{\delta(y_i, c) : i \in \mathcal{I}_b, c \in \mathbb{N}_m\}$ $\quad \in \{0,1\}^{n_b \times m}$

  $K_\theta \leftarrow \{k_\theta(x_i, x_j) : i \in \mathcal{I}_b, j \in \mathcal{I}_b\}$ $\quad \in \mathbb{R}^{n_b \times n_b}$

  $L_{\theta,\lambda} \leftarrow \text{cholesky}(K_\theta + n_b \lambda I_{n_b})$ $\quad \in \mathbb{R}^{n_b \times n_b}$

  $V_{\theta,\lambda} \leftarrow L_{\theta,\lambda}^T \backslash (L_{\theta,\lambda} \backslash Y)$ $\quad \in \mathbb{R}^{n_b \times m}$

  $P_{\theta,\lambda} \leftarrow K_\theta V_{\theta,\lambda}$ $\quad \in \mathbb{R}^{n_b \times m}$

  $r(\theta, \lambda) \leftarrow \alpha(\theta) \sqrt{\text{trace}(V_{\theta,\lambda}^T K_\theta V_{\theta,\lambda})}$

  $q(\theta, \lambda) \leftarrow \frac{1}{n_b} \sum_{i=1}^{n_b} \mathcal{L}_\epsilon((Y)_i, (P_{\theta,\lambda})_i) + 4e\, r(\theta, \lambda)$

  $(\theta, \lambda) \leftarrow \text{GradientBasedUpdate}(q, \theta, \lambda; \eta)$

until maximum iterations reached

Output: kernel hyperparameters θ, regularisation hyperparameter λ

Method

Idea: Consider the CME in the multiclass setting, referred to as the multiclass conditional embedding (MCE), whose empirical form is:

$$\hat{\mathbf{p}}(x) = \mathbf{f}(x) := \mathbf{Y}^T (K_\theta + n\lambda I)^{-1} \mathbf{k}_\theta(x). \tag{1}$$

Use Rademacher complexity bounds (RCB) to bound its expected risk:

Theorem 4.1 *For any $n \in \mathbb{N}_+$ and observations $\{x_i, y_i\}_{i=1}^n$ used to define $\mathbf{f}_{\theta,\lambda}$ (1), 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 $\mathbb{E}[\mathcal{L}_{e^{-1}}(Y, \mathbf{f}_{\theta,\lambda}(X))] \leq \frac{1}{n} \sum_{i=1}^n \mathcal{L}_\epsilon(Y_i, \mathbf{f}_{\theta,\lambda}(X_i)) + 4e\, r(\theta, \lambda) + \sqrt{\frac{8}{n} \log \frac{2}{\beta}}$, where the RCB is $r(\theta, \lambda) := \sqrt{\sup_{x \in \mathcal{X}} k_\theta(x, x) \text{tr}(\mathbf{Y}^T (K_\theta + n\lambda I)^{-1} K_\theta (K_\theta + n\lambda I)^{-1} \mathbf{Y})}$.*

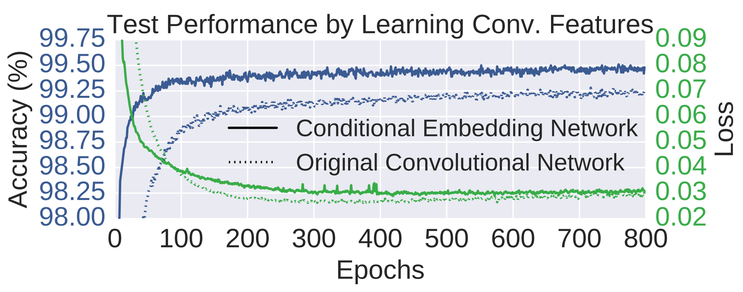
Propose an objective based on this bound, and extensions thereof, to ensure good generalization by balancing data fit and model complexity:

$$q(\theta, \lambda) := \frac{1}{n} \sum_{i=1}^n \mathcal{L}_\epsilon(y_i, \mathbf{f}_{\theta,\lambda}(x_i)) + 4e\, r(\theta, \lambda). \tag{2}$$

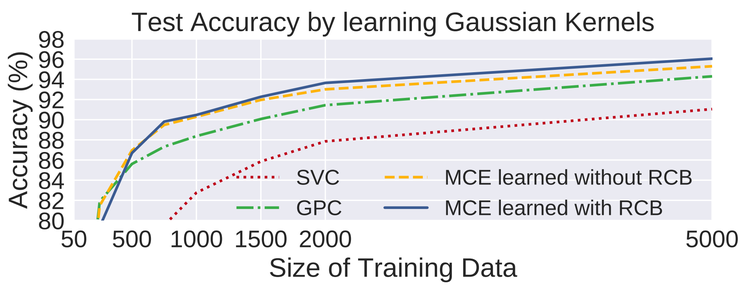
Deep MNIST & ARD MNIST

Deep MNIST: Apply our learning algorithm to an MCE with kernels constructed from deep convolutional neural networks. **Result**:

- Highly scalable with flexible representations
- Improved test accuracy: 99.48% v.s. 99.26%
- Faster convergence in network training



ARD MNIST: Apply our learning algorithm to perform automatic relevance determination (ARD) on MNIST pixels. **Result**:



UCI Experiments

Experimental Setup: Compare our learning algorithm to exisiting hyperparameter tuning algorithms on UCI datasets, as well as to other models as standard benchmarks. GMCE, GMCE-SGD, and CEN-1/2 are variations to our approach. GMCE and GMCE-SGD use anisotropic Gaussian kernels with full and batch stochastic gradient update. CEN-1 and CEN-2 employ kernels constructed from fully connected neural networks with 16-32-8 and 96-32 hidden units.

Result: Our learning algorithm achieves higher test accuracy across a range of datasets compared to other methods such as empirical risk minimization (ERM), cross validation (CV), and median heuristic (MED). In terms of the comparison to benchmark models, our algorithm performs on par with benchmarks using neural networks (a, c), probabilistic binary trees (b), decision trees (d), and regularized discriminant analysis (e).

Table 1: Test accuracy (%) of multiclass conditional embeddings on UCI datasets against benchmarks

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