Sample-based Approximate Regularization

Philip Bachman[1] PHIL.BACHMAN@GMAIL.COM
Amir-massoud Farahmand[1,2] AMIRMF@ANDREW.CMU.EDU
Doina Precup[1] DPRECUP@CS.MCGILL.CA
[1]School of Computer Science, McGill University, Montreal, Canada [2]Carnegie Mellon University, Pittsburgh, USA

Abstract

We introduce a method for regularizing linearly parameterized functions using general derivative-based penalties, which relies on sampling as well as finite-difference approximations of the relevant derivatives. We call this approach sample-based approximate regularization (SAR). We provide theoretical guarantees on the fidelity of such regularizers, compared to those they approximate, and prove that the approximations converge efficiently. We also examine the empirical performance of SAR on several datasets.

1. Introduction

Regularization is critical to controlling the complexity of hypothesis spaces and achieving favourable bias-variance trade-offs. Some machine learning methods even owe most of their success to an effective use of regularization. For example, a major reason for the success of SVMs is arguably their use of regularization that is "natural" in the RKHS tied to their kernel. For some choices of kernels, this Tikhonov-like regularization has a smoothness-inducing interpretation (Schölkopf & Smola, 2002). For example, the RKHS norm induced by the popular Gaussian RBF kernel penalizes all orders of derivatives of the learned function (Yuille & Grzywacz, 1988). Spline-based methods, which are ubiquitous in statistics but less common in machine learning, also rely on smoothness-inducing, derivative-based penalties. In particular, for univariate inputs or additive models, a second-order derivative penalty can be applied exactly in the nonparametric setting, leading to cubic smoothing splines (Wahba, 1990). But, this exact penalty quickly becomes intractable as the training set grows or the order of modeled interactions increases. While attempts have been made to produce computationally-efficient approximations of spline-like penalties (Eilers & Marx, 1996; Wood, 2003), full spline-based methods generally scale unfavorably.

In this paper, we introduce a method for efficiently approximating a general class of derivative-based penalties, which we call Sample-based Approximate Regularization (SAR). This approach applies to hypothesis spaces that are linearly parameterized, i.e., in which the input x is transformed into a feature space $\phi(x)$ and the output is approximated by $\phi(x)^\top \mathbf{w}$. This type of hypothesis space includes SVM-style approximators, feedforward neural networks, and various other types of regressors using features that are useful for particular application domains, such as SIFT, MFCC, etc (Lowe, 1999; Davis & Mermelstein, 1980). Based on the success of derivative-based penalties in the related RKHS and spline settings (Pearce & Wand, 2006), and on the empirical success of problem-specific features, it is desirable to obtain derivative-based regularizers that work with a wide range of feature transformations, e.g., ones not restricted to explicitly-computable RKHS kernels.

The SAR method can be used to augment the standard l_2 and l_1 regularizers that are commonly used with "general" feature transforms (i.e., non-kernel transforms). Conveniently, the regularizers produced by SAR are of the Tikhonov-type (i.e., $J(f_\mathbf{w}) = \mathbf{w}^\top \Sigma \mathbf{w}$ for some Σ), and can thus be applied efficiently with standard software. The computational complexity of SAR depends only loosely on the complexity of ϕ and not at all on the size of the training set, thus improving on costs of spline-based approaches to regularizing derivatives. We prove that the regularizers produced by SAR converge efficiently to the exact penalties they approximate. We also compare the loss of a SAR-regularized regression estimator to the loss of an estimator shaped by the exact regularizer approximated by SAR.

In the rest of this paper, we present our generalization of smoothness-inducing, derivative-based regularizers (Section 2), present our approach for approximating them efficiently (Section 3), analyze its theoretical properties (Section 4), and present empirical results illustrating the power of the proposed approach (Section 5).

Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32. Copyright 2014 by the author(s).

2. Smoothness-inducing regularizers

Consider a function $f : \mathcal{X} \to \mathbb{R}$ and a measure $\nu \in \mathcal{M}(\mathcal{X})$. If $f'(x)$ exists and is $L_2(\nu)$-integrable, then for $\mathcal{X} = \mathbb{R}$ a natural measure of the smoothness of f is:

$$\int_{\mathcal{X}} |f'(x)|^2 \mathrm{d}\nu \,, \tag{1}$$

One typically extends (1) to multi-dimensional domains $\mathcal{X} = \mathbb{R}^d$ by integrating the squared norm of the gradient. We propose instead a more general form, expressible as:

$$J_1(f) = \int_{\mathcal{X}} \oint_{S_x} (s^\top \nabla f)^2 \mathrm{d}s_x \mathrm{d}\nu \,. \tag{2}$$

The inner integral is over the surface S_x of a hyper-sphere centred at x, according to a location-dependent measure over directions $s_x \in \mathcal{M}(S_x)$. Each $s \in S_x$ corresponds to a unit-length vector pointing away from x in some direction. If s_x is set to a uniform distribution over the unit hyper-sphere for all $x \in \mathcal{X}$, then $J_1(f)$ is proportional to the integrated squared norm of $\nabla_x f$, due to the linearity of the dot-product. If s_x is the set of delta functions on the coordinate vectors of $\mathcal{X}$, $J_1(f)$ penalizes the integrated squared norm of $\nabla_x f$ exactly, as in the typical multi-dimensional extension of (1). But, the generalized derivative penalty $J_1(f)$ allows flexibility in assigning location-dependent importance to the variation of f along particular directions. Moreover, as we will see, it is amenable to sample-based approximation.

Another reasonable measure of the smoothness of f uses its second-order derivatives. In one dimension, if $f''(x)$ exists and is $L_2(\nu)$-integrable, then

$$\int_{\mathcal{X}} |f''(x)|^2 \mathrm{d}\nu \tag{3}$$

gives the standard penalty used in, e.g., cubic smoothing splines (Wahba, 1990). We extend the penalty in (3) to multiple dimensions as follows:

$$J_2(f) = \int_{\mathcal{X}} \oint_{S_x} \left(s^\top (H_x f) s\right)^2 \mathrm{d}s_x \mathrm{d}\nu \,, \tag{4}$$

where $s \in S_x$ are again distinct unit-length vectors jointly covering all directions pointing away from x, and $H_x f$ is the Hessian of f evaluated at x. When s_x is uniform over S_x, $J_2(f)$ penalizes the squared Frobenius norm $||H_x f||_F^2$ w.r.t. ν, which provides regularization that has proven useful in previous work (Rifai et al., 2011; Kim et al., 2009). As with (2), the generalized form in (4) encompasses a broad range of regularizers, due to flexibility in the choice of ν and s_x, and is amenable to approximation.

3. The SAR Method

The goal of our approach is to efficiently approximate regularizers of the form (2) and (4). The functionals J_1 and J_2 both involve integrating some quantity w.r.t. ν and s_x. SAR approximates the integrands in J_1 and J_2 efficiently using finite-difference approximations of directional derivatives and estimates the integrals using a Monte-Carlo approach based on samples from ν and s_x. We call methods to sample from ν *point samplers* and methods to sample from s_x *direction samplers*.

Algorithm 1 SAR($\tilde{p}_x, \tilde{p}_{s_x}, N, \phi, i, \epsilon$)

```
1: $\tilde{\Sigma}_i$ := zero matrix of size $p \times p$.
2: for $j = 1$ to $N$ do
3:     Sample $X_j$ from $\tilde{p}_x$.
4:     Sample $S_j$ from $\tilde{p}_{s_{X_j}}$.
5:     Compute $\delta_i^\epsilon(X_j, S_j, \phi)$    (see: (9) for defn.)
6:     $\tilde{\Sigma}_i := \tilde{\Sigma}_i + \delta_i^\epsilon(X_j, S_j, \phi)\delta_i^\epsilon(X_j, S_j, \phi)^\top$.
7: end for
8: return $\frac{1}{N}\tilde{\Sigma}_i$.
```

We focus on linearly-parameterized functions $f_{\mathbf{w}}(x) = \phi(x)^\top \mathbf{w}$, where $\phi : \mathcal{X} \to \mathbb{R}^p$ is a fixed feature transform, whose components are one-dimensional measurable functions $\{\varphi_i\}_{i=1}^p$, and $\mathbf{w} \in \mathbb{R}^p$ is a parameter vector. We denote the function space defined by the span of ϕ as $\mathcal{F}$.

Given a point sampler $\tilde{p}_x$, a direction sampler $\tilde{p}_{s_x}$, a sample size N, and a derivative order i, Algorithm 1 produces a matrix $\tilde{\Sigma}_i$ that defines SAR with: $\tilde{J}_i(f_{\mathbf{w}}) = \mathbf{w}^\top \tilde{\Sigma}_i \mathbf{w}$ for functions $f_{\mathbf{w}} \in \mathcal{F}$. To simultaneously regularize multiple derivative orders, their corresponding $\tilde{\Sigma}_i$ can be combined via element-wise summation.

Once an approximate regularizer $\tilde{\Sigma}_i$ has been produced by SAR, any method for estimating Tikhonov-regularized linear models can be applied. The computational cost of SAR comes from lines 3-6 of Algorithm 1. Assuming efficient point/direction samplers $\tilde{p}_x/\tilde{p}_{s_x}$, the feature extraction in line 5 and the outer products in line 6 dominate the cost of SAR. If the expected cost of computing $\phi(x)$ is c_ϕ, the target derivative order is i, and $\phi(x) \in \mathbb{R}^p$, then line 5 costs $c_\phi(i+1)$ per sample and line 6 costs p^2 per sample. Lines 3-6 each execute N times when using N samples to compute $\tilde{\Sigma}_i$. Depending on c_ϕ and p, either line 5 or 6 may dominate the overall cost. The discussion section further considers computation costs.

We now describe approaches for approximating the directional derivatives and for constructing samplers $\tilde{p}_x/\tilde{p}_{s_x}$.

3.1. Approximating directional derivatives

For functions $f_{\mathbf{w}} \in \mathcal{F}$, the first-order forward finite difference is given by $\frac{1}{\epsilon}\mathbf{w}^\top(\phi(x + \epsilon s) - \phi(x))$, thus:

$$\langle \nabla_x f_{\mathbf{w}}, s \rangle^2 \approx \mathbf{w}^\top \delta_1^\epsilon(x, s, \phi)\delta_1^\epsilon(x, s, \phi)^\top \mathbf{w}, \tag{5}$$

in which we introduce the first-order *finite difference vector* $\delta_1^\epsilon(x, s, \phi) \triangleq \frac{1}{\epsilon}(\phi(x + \epsilon s) - \phi(x))$.

For a point x and direction s, the term $s^\top(H_x f)s$ in (4) is equivalent to the second-order directional derivative of f, at x, in direction s, with finite difference approximation:

$$s^\top(H_x f)s \approx \frac{f(x) - 2f(x + \epsilon s) + f(x + 2\epsilon s)}{\epsilon^2} \qquad (6)$$

For $f_\mathbf{w} \in \mathcal{F}$, the square of this term is given by:

$$(s^\top(H_x f_\mathbf{w})s)^2 \approx \mathbf{w}^\top \delta_2^\epsilon(x, s, \phi)\delta_2^\epsilon(x, s, \phi)^\top \mathbf{w}, \qquad (7)$$

in which we use the second-order finite difference vector $\delta_2^\epsilon(x, s, \phi) \triangleq \frac{1}{\epsilon^2}(\phi(x) - 2\phi(x + \epsilon s) + \phi(x + 2\epsilon s))$.

Based on (5) and (7), the key to SAR is that the integrals in J_1 and J_2 can be approximated by:

$$\mathbf{w}^\top \left(\int_\mathcal{X} \oint_S \delta_i^\epsilon(x, s, \phi)\delta_i^\epsilon(x, s, \phi)^\top \mathrm{d}s_x \, \mathrm{d}\nu \right) \mathbf{w}, \qquad (8)$$

in which we use finite difference vectors and $i \in \{1, 2\}$ indicates the derivative order to regularize.

To regularize higher-order derivatives with SAR, only the finite difference vectors used in (8) need to change:

$$\delta_i^\epsilon(x, s, \phi) = \sum_{j=0}^{i} (-1)^j \binom{i}{j} \frac{\phi\left(x + (i - j)\epsilon s\right)}{\epsilon^i}. \qquad (9)$$

When regularizing a single order i with fixed ϵ, the denominator ϵ^i in (9) can be ignored, as it is constant for all δ_i^ϵ. In this case, numerical precision (for $\epsilon^i \to 0$) is not an issue. A similar idea can be applied when regularizing across multiple orders. Principled approaches to select ϵ and minimize the side-effects from finite precision are subject for future work. We note that we have not run into any numerical problems in the experiments.

3.2. Sampling from ν and s_x

We now describe concrete methods to sample from ν and s_x. Suppose we are given a set $\mathcal{D}_n = \{X_1, X_2, \ldots, X_n\}$ of "training" input observations $X_i \in \mathbb{R}^d$, drawn from the source distribution $p(x)$. We will approximate ν using N samples, contained in a set $\mathcal{D}'_N$.[1] A natural choice for the sampler is to draw values from an approximation to $p(x)$ obtained by perturbing the existing points $\mathcal{D}_n$, an approach based on the manifold/cluster assumption underlying most work on semi-supervised learning. Let $\mathcal{L}$ be a distribution over lengths, which determines the degree of "smoothing" to apply during sampling. Algorithm 2 samples from the empirical approximation to $p(x)$, convolved with the isotropic distribution with length distribution $\mathcal{L}$.

[1] In the supervised learning setting, $\mathcal{D}_n$ contains label information as well, but we ignore it in the process of generating $\mathcal{D}'_N$. In a semi-supervised setting, we can also use unlabelled data.

Algorithm 2 FuzzyPointSampler($\mathcal{D}_n$, N, $\mathcal{L}$).

```
1: for $j = 1$ to $N$ do
2:     Sample $X_j$ from $\mathcal{D}_n$ uniformly at random.
3:     Sample a direction $S_j$ uniformly at random.
4:     Sample a perturbation length $\epsilon_j$ from $\mathcal{L}$.
5:     Add $\tilde{X}_j = X_j + \epsilon_j S_j$ to $\mathcal{D}'_N$.
6: end for
7: return $\mathcal{D}'_N$.
```

Algorithm 3 BlurryBoxSampler($\mathcal{D}_n$, N, $\mathcal{L}$)

```
1: Compute the minimal bounding box for the $\mathcal{D}_n$.
2: for $j = 1$ to $N$ do
3:     Sample $X_j$ uniformly from within the box.
4:     Sample a direction $S_j$ uniformly at random.
5:     Sample a step length $\epsilon_j$ from $\mathcal{L}$.
6:     Add $\tilde{X}_j = X_j + \epsilon_j S_j$ to $\mathcal{D}'_N$.
7: end for
8: return $\mathcal{D}'_N$.
```

The second method samples approximately from the uniform distribution over $\mathcal{X}$, to mimic the distribution implicit in the RKHS regularization accompanying Gaussian RBFs. Algorithm 3 samples from a uniform distribution over the smallest axis-aligned box enclosing D_n convolved with the isotropic distribution with length distribution $\mathcal{L}$.

We propose two methods to sample directions from s_x. The first is to sample a unit direction uniformly at random. The second is to sample a unit direction uniformly at random, transform it by some matrix, and then rescale it to unit length. The first method produces a regularizer that penalizes derivatives in all directions equally, and the second biases the penalty based on the eigenvectors and eigenvalues of the transform matrix. Developing direction samplers with location-dependent biases, e.g., to emphasize invariance w.r.t. small translations/rotations in an object recognition task, is an interesting topic for future work.

4. Theoretical Analysis

The goal of this section is twofold. First, we study the behaviour of a SAR-based regularized least-squares regression estimator (Theorem 1). Second, we focus on the convergence behaviour of the sample-based approximate regularizer $\tilde{J}_N(\cdot)$ to the regularizer $J(f)$. We provide two results, one in the form of the supremum of the empirical process (Theorem 2) and the other in the form of the supremum of the modulus of continuity of the empirical process (Theorem 3). For simplicity, we only study the 1st-order derivative-based regularizer and its central difference-based SAR.

Let us first define some notations. The gradient of a function $f : \mathbb{R}^d \to \mathbb{R}$ is denoted by $\nabla f(x)$. We denote the central difference approximation of the gradient by $(\triangle_\varepsilon f)(x) = [(\triangle_\varepsilon f)_1(x) \cdots (\triangle_\varepsilon f)_d(x)]$ with $(\triangle_\varepsilon f)_i(x) = \frac{f(x+\varepsilon e_i)-f(x-\varepsilon e_i)}{2\varepsilon}$, where e_i are the unit coordinate vectors.

Given a probability distribution $\nu \in \mathcal{M}(\mathcal{X})$, the 1st-order derivative-based regularizer[2] is $J(f) = \int \|\nabla f(x)\|^2 \, \mathrm{d}\nu(x)$. Given $\mathcal{D}'_N = \{X'_1, \dots, X'_N\}$ with $X'_i \overset{\text{i.i.d.}}{\sim} \nu$, we define the sample-based approximate regularizer as: $\tilde{J}_N(f) = \frac{1}{N}\sum_{i=1}^N \|\triangle_\varepsilon f(X'_i)\|^2$. We also define $J_N(f) = \frac{1}{N}\sum_{i=1}^N \|\nabla f(X'_i)\|^2$. Note that for $f_{\mathbf{w}} \in \mathcal{F}$, we have $J(f_{\mathbf{w}}) = \mathbf{w}^\top \Sigma \mathbf{w}$ with the true Grammian $\Sigma \triangleq \int \sum_{i=1}^d \frac{\partial \phi(x)}{x_i} \frac{\partial \phi(x)}{x_i}^\top \mathrm{d}\nu(x)$. Similarly, we have $\tilde{J}_N(f_{\mathbf{w}}) = \mathbf{w}^\top \tilde{\Sigma}_N \mathbf{w}$ with the approximate empirical Grammian $\tilde{\Sigma}_N \triangleq \frac{1}{N}\sum_{i=1}^N \sum_{j=1}^d (\triangle_\varepsilon \phi)_j^\top (X_i)(\triangle_\varepsilon \phi)_j(X_i)$.[3] For a fixed $L > 0$, the truncation operator $\beta_L : \mathcal{F} \to \mathcal{F}$ is defined as $(\beta_L f)(x) \triangleq f(x)$ when $|f(x)| \le L$ and $(\beta_L f)(x) \triangleq \mathrm{sgn}\big(f(x)\big)L$ otherwise.

The regression setup is as follows. Let $\mathcal{D}_n = \{(X_i, Y_i)\}_{i=1}^n$ be a dataset with $X_i \overset{\text{i.i.d.}}{\sim} \mu$. Assume that the probability distribution generating the data is such that $|Y| \le L$ (almost surely) with $L > 0$. Denote by $f^*(x) = \mathbb{E}[Y|X=x]$ the regression function, which in general does not belong to $\mathcal{F}$. Given $\mathcal{D}_n$ and an independent dataset $\mathcal{D}'_N$, the SAR-based regression estimator $\hat{f}_n$ is defined as the L-truncated estimator $\hat{f}_n \triangleq \beta_L \bar{f}_n$, with

$$\bar{f}_n \leftarrow \operatorname*{argmin}_{f \in \mathcal{F}} \frac{1}{n}\sum_{i=1}^n (f(X_i) - Y)^2 + \lambda \tilde{J}_N(f). \qquad (10)$$

We now provide an upper bound on the performance of this estimator. To state our result, for $k \ge 1$, we define

$$D_k(\phi) \triangleq \max_{i=1,\dots,d} \sup_{x \in \mathcal{X}} \left\| \frac{\partial^k \phi(x)}{\partial x_i^k} \right\|_2 .$$

If the k-th partial derivatives are not defined, we set $D_k(\phi) = \infty$. For our results, we require the existence of $D_1(\phi)$ and $D_3(\phi)$. All proofs are deferred to the Supplementary Material.

[2]If $\mathcal{X}$ is a proper open subset of $\mathbb{R}^d$, for some samples X' close to the boundary of $\mathcal{X}$, $(\triangle_\varepsilon f)_i(X')$ may not be defined (because one side can be outside the domain). If we ensure that supp(ν) is at least ε away from the boundary in the l_∞-norm, all the results hold with $\mathcal{X} \subset \mathbb{R}^d$ instead of $\mathcal{X} = \mathbb{R}^d$.

[3]Note that the meaning of subscripts of J and $\tilde{J}$ is different from Section 3. Here J_N and $\tilde{J}_N$ refer to the use of N samples to estimate the 1st-order derivative (using the true derivative or its finite difference approximation, respectively), while in the previous section we used J_i and $\tilde{J}_i$ to refer to the ith-order derivative-based regularizer and its SAR version. No confusion should arise as we always use N to refer to the number of samples.

Theorem 1. *Assume that all $\{\varphi_i\}_{i=1}^p$ are three-time differentiable and $\sup_{x \in \mathcal{X}} \|\phi(x)\|_2 \le R$. Moreover, suppose that $\lambda_{\min}(\tilde{\Sigma}_N)$, the smallest eigenvalue of $\tilde{\Sigma}_N$, is bounded away from zero. There exist constants $c_1, c_2 > 0$ such that for any fixed $\delta > 0$, with probability at least $1-\delta$, we have:*

$$\int \left|\hat{f}_n(x) - f^*(x)\right|^2 \mathrm{d}\mu(x) \le$$
$$2 \min_{\mathbf{w} \in \mathbb{R}^p, \|f_{\mathbf{w}}\|_\infty \le L} \Bigg\{ 2\int |f_{\mathbf{w}} - f^*(x)|^2 \mathrm{d}\mu(x) + 2\lambda J(f_{\mathbf{w}}) +$$
$$\lambda \|\mathbf{w}\|_2^2 \, d\Big(\frac{8 D_1^2(\phi)\log(3/\delta)}{3N} +$$
$$\frac{\varepsilon^2}{6} D_3(\phi)[2D_1(\phi) + \frac{\varepsilon^2}{6} D_3(\phi)]\Big)\Bigg\}$$
$$+ \frac{c_1 L^6 R^2}{\lambda_{\min}(\tilde{\Sigma}_N)} \frac{\log(nL)}{n\lambda} + \frac{c_2 L^4 \log(1/\delta)}{n}.$$

This result shows the effects of function approximation and estimation errors, the way regularization coefficient λ and $J(f_{\mathbf{w}})$ determine their tradeoff, and the error caused by SAR. The term $\min_{\mathbf{w} \in \mathbb{R}^p} \int |f_{\mathbf{w}} - f^*(x)|^2 \mathrm{d}\mu(x) + \lambda J(f_{\mathbf{w}})$ is the [regularized] approximation error and indicates how well the target function f can be approximated in a subset of $\mathcal{F}$. The subset is determined by the true regularization functional $J(f_{\mathbf{w}}) = \mathbf{w}^\top \Sigma \mathbf{w}$ and λ. As usual in regularized estimators, increasing λ might increase the approximation error, but it decreases the estimation error $O(\frac{\log(n)}{n\lambda})$ on the other hand, and vice versa. If $\mathcal{F}$ as defined by the basis functions "matches" the target function (i.e., f^* can be well-approximated with a function in $\mathcal{F}$ with a small $J(f)$), we can learn the target function fast. This is how feature-engineering or data-dependent feature generation show their benefits. It is noticeable that this result does not depend on the dimension of the feature space p.

Results similar to this part of the theorem are known in the supervised learning literature, cf. Theorem 21.1 of Györfi et al. (2002) for regularized regression in $C^k(\mathbb{R})$ (splines), Theorem 7.23 of Steinwart & Christmann (2008) for regularized loss in an RKHS, and Sridharan et al. (2009) for strongly convex objectives (which is satisfied for a convex loss and the l_2 regularizer) and linear function spaces.

The effect of using $\tilde{J}_N(f)$ instead of the true regularizer $J(f)$ in (10) appears in the $O(\|\mathbf{w}\|_2^2 [\frac{1}{N} + \varepsilon^2])$ term. The curious observation here is that the effect depends on the size of $\mathbf{w}$, so if the true function can be well-approximated by a "simple" function (measured according to $\|\mathbf{w}\|_2$), we would not suffer much from the error caused by SAR.

To better understand the behaviour of the bound, consider the case that $J(f_{\mathbf{w}}) = \|\mathbf{w}\|_2^2$ and the target function f^* belongs to $\mathcal{F}$, i.e., $f^* = f_{\mathbf{w}^*}$ for some $\mathbf{w}^*$. Ignoring the constants and the logarithmic terms, by choosing $\lambda = \frac{1}{\|\mathbf{w}^*\|\sqrt{n}}$ to optimize the tradeoff between $\lambda J(f^*_{\mathbf{w}})$ and $\frac{1}{n\lambda}$, we get the upper bound of $O(\frac{\|\mathbf{w}^*\|_2}{\sqrt{n}}[1+\frac{1}{N}+\varepsilon^2])$.

Remark 1. One could get $\int |f_\mathbf{w} - f^*(x)|^2 \mathrm{d}\mu(x) + \lambda J(f_\mathbf{w})$ inside the minimizer instead of the current one, which has a multiplicative constant of 2, at the price of having $O(\frac{\|\mathbf{w}\|_2^2}{\sqrt{N}}+\frac{1}{\sqrt{n}})$ instead of $O(\frac{\|\mathbf{w}\|_2^2}{N}+\frac{1}{n})$. This depends on whether we use Bernstein's inequality or Hoeffding's inequality in the proofs.

Remark 2. The quantity $\lambda_{\min}(\tilde{\Sigma}_N)$ in the theorem is a random function of $\mathcal{D}'_N$ and can be calculated given $\mathcal{D}'_N$.

We now depart from the context of regression and focus on the SAR procedure itself. The first result is a uniform upper bound on the difference between $J(f)$ and $\tilde{J}_N(f)$ for any function $f \in \mathcal{F}_B \triangleq \left\{ \phi^\top \mathbf{w} : \mathbf{w} \in \mathbb{R}^p, \|\mathbf{w}\|_2 \le B \right\}$, i.e., the ball with radius B w.r.t. the l_2-norm of $\mathbf{w}$.

Theorem 2 (Supremum of the Empirical Process $|\tilde{J}_N(f) - J(f)|$)**.** *Assume that all $\{\varphi_i\}_{i=1}^p$ are three-time differentiable. For any fixed $\delta > 0$ and $B > 0$, we have:*

$$\sup_{f\in\mathcal{F}_B}\left|\tilde{J}_N(f) - J(f)\right| \le \frac{B^2\varepsilon^2}{6} d D_3(\phi)\left(2D_1(\phi) + \frac{\varepsilon^2}{6}D_3(\phi)\right) + 32B^2 d D_1^2(\phi)\sqrt{\frac{2p\log\left(\frac{128B^2 d D_1^2(\phi)N}{\delta}\right)}{N}} + \frac{1}{N},$$

with probability at least $1-\delta$.

This theorem shows the effects of the estimation error and the finite difference approximation error. The simplified behaviour of the estimation error is $O(B^2\sqrt{\frac{p}{N}})$. The dependence on N and p is common to the usual uniform deviation bounds in statistical learning for functions from a p-dimensional linear vector space. The effect of the size of the function space also manifests itself through B^2.

The effect of the finite difference approximation error is $O(B^2\varepsilon^2)$ – neglecting terms depending on the smoothness of the basis functions. The ε^2 dependence is the usual residual error from the central difference approximation of a derivative. If instead we used a forward (or backward) estimate of the derivative, we would get ε behaviour. The dependence on B is because functions $\phi^\top\mathbf{w}$ with larger $\|\mathbf{w}\|_2$ might have a larger derivatives, so their finite difference approximation would have a larger residual error.

Theorem 2 provides an upper bound for the supremum of the empirical process only over a subset $\mathcal{F}_B$ of $\mathcal{F}$, but it does not provide a non-trivial result for the supremum of $|\tilde{J}_N(f) - J(f)|$ over $\mathcal{F}$. This is expected as for large $\mathbf{w}$, the true regularizer $J(f_\mathbf{w})$ would be large too, and the deviation of $\tilde{J}_N(f_\mathbf{w})$ around it can also be large. Nonetheless, we can still study the behaviour of the empirical process as a function of $J(f)$. This is known as the modulus of continuity result in the empirical process theory (or relative deviation of error). The following theorem provides such a result. Here we denote $a \vee b = \max\{a, b\}$.

Theorem 3 (Modulus of Continuity for the Empirical Process $|\tilde{J}_N(f) - J(f)|$)**.** *Assume that all $\{\varphi_i\}_{i=1}^p$ are three-time differentiable. Suppose that $\lambda_{\min}(\Sigma)$, the smallest eigenvalue of Σ, is bounded away from zero. W.l.o.g., assume that $256dD_1^2(\phi) \ge 1$. Let $\alpha > 0$. There exists $c_1(\alpha)$ such that for any fixed $\delta > 0$, we have*

$$\sup_{f\in\mathcal{F}}\frac{\left|\tilde{J}_N(f) - J(f)\right|}{[J(f)\vee\lambda_{\min}(\Sigma)]^{1+\alpha}} \le \frac{1}{\lambda_{\min}^{1+\alpha}(\Sigma)}\left[2^5 d D_1^2(\phi)\sqrt{\frac{2p\log\left(\frac{512dD_1^2(\phi)c_1(\alpha)N}{\delta}\right)}{N}} + \frac{d\varepsilon^2}{3!}D_3(\phi)\left[2D_1(\phi) + \frac{\varepsilon^2}{3!}D_3(\phi)\right]\right],$$

with probability at least $1-\delta$. Here $c_1(\alpha)$ can be chosen as follows: For $0 < \alpha \le \frac{1}{4e\log(2)} \approx 0.1327$, $c_1(\alpha) = 8[2 - \frac{W_{-1}(-4\alpha\log(2))}{4\alpha\log(2)}]$ (in which W_{-1} is the lower branch of Lambert W-function), and $c_1(\beta) = 16$ otherwise.

We can elucidate this result by seeing how it works in the context of Theorem 2, by restricting $\mathcal{F}$ to $\mathcal{F}_B$. In this case, $J(f) \le O(B^2)$, so we get $\sup_{f\in\mathcal{F}_B}|\tilde{J}_N(f) - J(f)| \le c_2(d, D_1, D_3, \Sigma)B^{2(1+\alpha)}[\sqrt{\frac{p\log(c_1(\alpha)N/\delta)}{N}} + \varepsilon^2]$ instead of $c_3(d, D_1, D_3)B^2[\sqrt{\frac{p\log(B^2N/\delta)}{N}} + \varepsilon^2]$ in Theorem 2. The major difference is in the exponent of B. When α goes to zero, $B^{2(1+\alpha)}$ decreases, but the term $c_1(\alpha)$ inside the logarithm increases. As can be seen from the definition of $c_1(\alpha)$, when $\alpha \to 0$, $c_1(\alpha)$ blows up. Overall, even though Theorem 2 provides a slightly tighter upper bound on the error for $\mathcal{F}_B$, Theorem 3 can be considered a stronger result as it holds for all functions in $\mathcal{F}$.

Remark 3. The effect of the input space dimension d on SAR's statistical properties, as can be seen in all results, is quite mild, and only appears in constants. SAR's sampling is a typical Monte Carlo integration, for which convergence rate is dimension-independent. The minor effect of d is due to using finite differences and the way we have defined D_k.

Finally it is worth mentioning that in the manifold regularization literature, there are results similar to Theorem 2. In particular, they provide conditions that the error between the various variants of the graph Laplacian-based and the Laplace-Beltrami-based regularizers goes to zero. For example, Bousquet et al. (2004) proved the asymptotic convergence for a fixed function. This should be compared to our much stronger uniform convergence rate over a function class – albeit the regularizers are different. Belkin & Niyogi (2008) showed the asymptotic uniform convergence over a class of functions, but did not provide a convergence rate. Hein (2006) extended that result and provided a convergence rate over a subset of Hölder-continuous functions.

In contrast to Theorem 1, the results in those papers did not consider the effect of error in regularization to the estimator (e.g., classifier or regression estimator), though Hein (2006) mentioned that his result could be used to prove consistency for algorithms that use graph Laplacian-based regularizers. This would be similar to using Theorem 2 to prove error bounds, which is a path that we did not take. In the different context of transductive learning (or semi-supervised learning on graphs), Belkin et al. (2004) provided a generalization error result for regularized algorithms on graphs, with a graph Laplacian-based regularizer being one possible choice, using tools from algorithmic stability. None of these papers provides a modulus of continuity result similar to Theorem 3.

5. Experiments

Our first tests involved least-squares regression with inputs $x \in \mathbb{R}$ and outputs $y \in \mathbb{R}$. The data distribution was designed to emphasize SAR's ability to regularize heterogenous basis functions. This contrasts with standard RKHS regularization, which uses more restricted collections. The joint distribution over (x, y) was set so four cycles of a sin wave occurred over the input domain, each with a wavelength 2.5 times longer than the previous one. The wave amplitude was scaled linearly from 1 to 2 over the input domain. The density of x was set so the expected number of observations seen for each cycle was the same. The training y values were corrupted by zero-mean Gaussian noise with standard deviation scaling linearly from 0.2 to 0.4 over the input domain. Performance was measured using uncorrupted y values. We call this distribution SynthSin1d.

The smooth sinusoid underlying SynthSin1d seems amenable to RKHS-regularized RBF regression, but causes problems due to large changes in the length scale of useful correlations over the input domain. When restricted to fixed bandwidth RBFs, the RKHS approach will always underperform on some part of the function not suited to the chosen bandwidth, as shown by results in Figure 1a.

Using SynthSin1d, we compared the performance of SAR2 regularization with L2 regularization and RKHS regularization of Gaussian RBFs. SAR2 and L2 regularization were applied to four RBFs anchored at each training point, with bandwidths $\gamma \in \{2, 4, 8, 16\}$. RKHS regularization was applied independently at each bandwidth, using the same RBFs, i.e., four RKHS-regularized solutions were learned for each train/test set. We compared the performance of the three methods on training sizes $\in [50...100]$. For each training size, 100 training sets were sampled from SynthSin1d (with output noise) and, for each set, the function learned with each regularizer was tested on 5000 points sampled from SynthSin1d (without output noise). Regularization weights for each method were set independently for each training size, to maximize measured performance.

We measured performance as the percentage of variance in the true function recovered by the learned approximation:

$$\% \text{ variance recovered} = 1 - \frac{\sum_i (\hat{y}_i - y_i)^2}{\sum_j (y_j - \bar{y})^2}, \qquad (11)$$

in which $\hat{y}_i$ gives the value of the learned approximation at test point x_i, y_i gives the value of the true function at x_i, and $\bar{y}$ gives the mean of the true function. The value of (11) approaches 1 as the approximation approaches the true function (i.e., larger values are better).

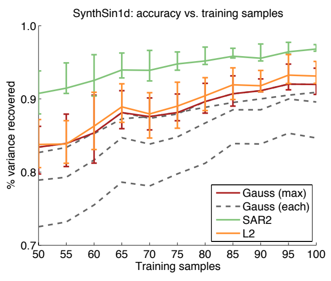
Figure 1a plots the mean performance of each regularization method for each considered training set size, with error bars indicating the upper and lower quartiles over the 100 tests at each size. The performance of RKHS regularization at each bandwidth is plotted in gray and the maximum performance is in red. In these tests, SAR2 significantly outperformed both L2 regularization using the same basis functions and RKHS regularization using any of the fixed-bandwidth subsets of the basis functions.

Our second tests extended the form of SynthSin1d to inputs $(x_1, x_2) \in \mathbb{R}^2$ and outputs $y \in \mathbb{R}$. We call this distribution SynthSin2d. Importantly, the value of y depended most strongly on x_1, making x_2 relatively uninformative. We performed 100 tests at each of the same training sizes as for SynthSin1d. SAR2 and L2 regularization were applied to collections of three Gaussian RBFs anchored at each training point, with bandwidths $\gamma \in \{0.5, 2, 8\}$. RKHS regularization was applied independently for each fixed-bandwidth RBF subset. Regularization weights for each method were set at each training size, to maximize measured performance. We also measured the performance of SAR2 regularization with direction sampling biased as follows: select a direction (x_1, x_2) uniformly, multiply its x_2 by 10, and then rescale (x_1, x_2) to the desired length. A SAR regularizer computed subject to this bias more severely penalizes change in the estimated function along the x_2 axis, which was known to be less informative.

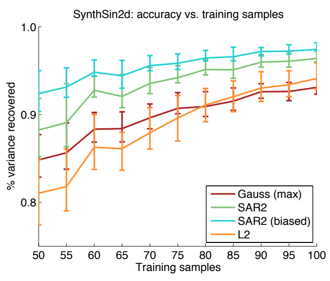
Figure 1b shows that SAR2 significantly improves on the performance of strong RKHS regularization applied to a more restricted set of basis functions and simple L2 regularization applied to an equally flexible set of basis functions. Adding a "correct" bias during regularizer construction further improves the advantage of SAR2, particularly for small training sets. Figure 1c/d qualitatively compares the behavior of L2 and biased SAR2 regularization. Biased SAR2 "interpolates" noticeably better than L2.

5.1. Natural Data

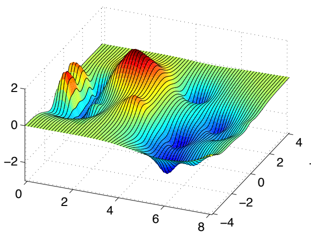
We write "full RBF" for RBFs based on the values of *all features* of an observation, and we write "univariate RBF"



(a)



(b)


(c) SynthSin2d: L2

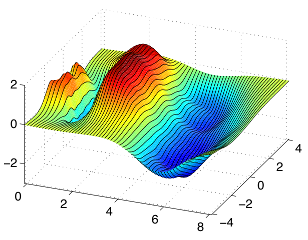

(d) SynthSin2d: SAR2+bias

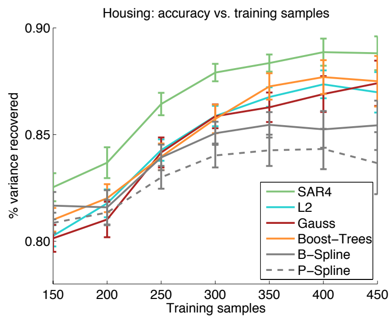
Figure 1. Full descriptions of the tests underlying these plots are given in the main text. (a) shows the performance of SAR2 on SynthSin1d when learning with Gaussian RBFs with multiple bandwidths, w.r.t. the number of training samples. We also plot the performance of L2 regularization applied to the same RBFs and the performance of RKHS-regularized regressions for each fixed-bandwidth subset of the RBFs. The best performance of RKHS regularization over the considered bandwidths is highlighted in red and per-bandwidth performances are plotted separately in gray. (b) is analogous to (a), but for tests based on SynthSin2d. (b) also plots the performance of SAR2 with biased directional sampling, which penalizes non-linearity in the learned function more along the axis which, for SynthSin2d, was uninformative. (c)/(d) compare the qualitative behavior of L2 and biased SAR2 on SynthSin2d.

for RBFs based on the value of *a single feature* of an observation. RBFs were Gaussian and RKHS regularization was applied during estimation, unless noted otherwise.

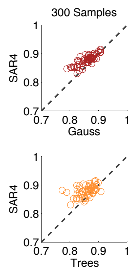
We tested SAR with the "Boston housing" dataset from UCI/StatLib, which comprises 506 observations $x \in \mathbb{R}^{13}$ describing features of neighborhoods in the Boston area (circa 1978), with the prediction target being the median value of homes in each neighborhood. We preprocessed the observations by setting features to zero mean and unit variance, and setting the targets to zero mean. We compared six methods: L2, SAR4, Gaussian RBFs, 4th-order B-spline RBFs, additive P-splines, and boosted Trees. We measured performance with respect to (11).

We performed tests with training sets of size 150-450. For each size, 100 rounds of randomized cross validation were performed, with non-training examples used for evaluating performance. When boosting trees, we set the maximum depth to 3 and performed 250 rounds of boosting with a shrinkage factor of 0.1, which maximized measured performance. For other methods, we set regularization weights separately for each training size to maximize measured performance. Kernel bandwidths were selected to maximize performance with 300 training samples.

L2, SAR4, and Gauss all used full Gaussian RBFs centered on each training point with bandwidth $\gamma = 0.05$ fixed across all tests. B-spline used 4th-order B-spline RBFs centered on each training point with bandwidth $\gamma = 0.2$. P-spline applied 4th-order regularization to 2nd-order additive B-spline bases with 30 knots per dimension. In addition to full RBFs, L2 and SAR4 used a collection of univariate RBFs, with the RBFs on each axis centered on the empirical deciles of the corresponding features. The standard deviation of each univariate RBF was set to the max-



(a)



(b)

Figure 2. (a) compares performance on the "Boston housing" data. Error bars give 95% confidence intervals. (b) shows per-round outcomes of each train/test round at training set size 300. y axes give accuracy of SAR4 and x axes give accuracy of RKHS-regularized Gaussian RBFs or boosted trees.

imum of the distances to its upper and lower "neighbors". The single binary feature in this dataset was represented by just two univariate RBFs, centered on its min/max values. Univariate RBF structure was not optimized.

SAR4 estimated approximate regularizers for first through fourth-order derivatives and combined the resulting matrices naively, by an unweighted sum. SAR4 used a compound point sampler which drew 75% of its samples from the fuzzy point sampler in Algorithm 2 and 25% of its samples from the blurry box sampler in Algorithm 3. Both samplers were constructed strictly from the training set during each round of CV. An unbiased direction sampler with stochastic lengths was used. The length distributions $\mathcal{L}$ in point/direction sampling were set to the non-negative half of a normal distribution, with standard deviation set to 0.5/0.2 times the median nearest-neighbor distance in the

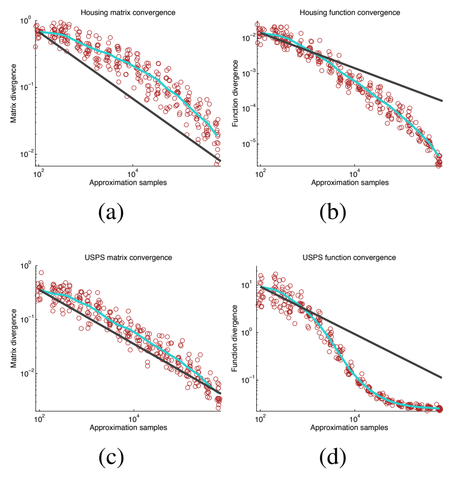


Figure 3. SAR4 convergence on housing and USPS data. Left column: convergence of the regularizer matrix. Right column: convergence of the learned function. Parameters for (a)/(b) were as before, using 50 training sets of size 300. USPS results in (c)/(d) used 50 random training sets of size 500. y axes in (a)/(c) give $\sup_{\mathbf{w}:||\mathbf{w}||_2\leq 1} |\tilde{J}_x(f_\mathbf{w}) - \tilde{J}_\infty(f_\mathbf{w})|$, which tracks convergence of the SAR-induced penalty. The lighter line plots empirical mean at each sample size and the darker line plots the theoretical rate $1/\sqrt{N}$. y axes in (b)/(d) measure difference between the function induced by a regularizer based on x samples and a converged one.

training set. A lower bound of 0.05 was set on the effective step length ϵ.[4] The sampler parameters were not optimized.

Figure 2 presents these tests. SAR4 consistently outperformed the other methods, as seen in 2a. Figure 2b examines relative performance more closely, by plotting results on individual train/test splits for training sets of size 300. SAR4 outperformed boosted trees and Gauss-RBF on most splits. Figure 3 examines SAR4's convergence in this setting.

Our final tests used the standard USPS/MNIST digit recognition datasets. We tested on 100 randomly sampled training sets of size 500/2500 and tested on points not seen in training. We compared standard L2, RKHS, and SAR4 regularization using sampler parameters matching those used for tests on the housing data. Each method used full Gaussian RBFs at each training point (as for an SVM), with bandwidth $\gamma = 0.015/0.025$, which were selected to maximize performance of RKHS regularization. We optimized the 1-vs-all squared hinge loss. Regularization weights were set to maximize measured performance. For L2/RKHS/SAR4 the mean and standard deviation of classification accuracy in these tests was 92.7(0.5)/94.0(0.4)/94.1(0.4) for USPS and 94.5(0.02)/95.2(0.01)/95.4(0.02) for MNIST. Both RKHS and SAR4 significantly outperformed L2 on USPS. All pairwise comparisons were significant on MNIST. Figure 3 illustrates convergence of SAR on the USPS data. Note that MNIST tests used $\phi(x) \in \mathbb{R}^{2500}$ for $x \in \mathbb{R}^{784}$.

[4]This was always much less than the st. dev. of $\mathcal{L}$.

6. Discussion

SAR provides a general approach to controlling complexity in a broad class of functions, i.e., those representable by linear combinations of a fixed set of basis functions, by minimizing the n^{th}-order derivative. For $n = 1$, we provided bounds on the error in the regularizer produced by SAR and showed that the approximation process is reasonably sample-efficient. The main benefit of SAR is its flexibility, as can be seen from the empirical examination.

Some other work in the manifold learning literature uses the data distribution to define data-dependent regularizers. For instance, Bousquet et al. (2004) defines a density-based regularizer. But, their practical implementation only considers a first-order derivative-based regularizer using Gaussian basis functions. SAR provides a more general framework to regularize higher-order derivatives, without requiring analytically tractable integrals.

When the data belongs to a low-dimensional manifold, a common choice is to use the norm of the Laplace-Beltrami operator on the manifold. However, this norm cannot be computed analytically in most cases, so sample-based approximations are used, e.g., the graph Laplacian operator Zhu et al. (2003); Belkin et al. (2006).[5] SAR is more general and is not designed with the goal of approximating the Laplace-Beltrami-based regularizer.

SAR raises a number of other interesting questions. On the theoretical side, it would be interesting to analyze SAR for higher-order derivatives, establish the influence of structure in the point measure ν and direction measure s_x, or make precise the relation between SAR and Laplacian-based regularization. On the practical side, developing heuristic approaches to reduce the effective sample complexity, as well as point and direction samplers that better leverage prior knowledge is desirable. Reducing the per-sample cost of SAR by leveraging techniques for reduced-rank kernel approximation in SVMs and implementing SAR so as to take advantage of sparsity in $\phi(x)$ both seem worthwhile, as they could significantly reduce the cost of the outer-products in line 6 of Algorithm 1.

Acknowledgments

This work was supported by NSERC.

[5] As discussed by Nadler et al. (2009), the use of the first-order derivative is not appropriate for high-dimensional input spaces, so one might use higher-order derivatives instead (Kim et al., 2009; Zhou & Belkin, 2011).

References

Belkin, Mikhail and Niyogi, Partha. Towards a theoretical foundation for laplacian-based manifold methods. 74(8): 1289–1308, 2008. 5

Belkin, Mikhail, Matveeva, Irina, and Niyogi, Partha. Regularization and semi-supervised learning on large graphs. In Shawe-Taylor, John and Singer, Yoram (eds.), *COLT*, volume 3120 of *Lecture Notes in Computer Science*, pp. 624–638. Springer, 2004. 6

Belkin, Mikhail, Niyogi, Partha, and Sindhwani, Vikas. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research (JMLR)*, 7:2399–2434, 2006. 8

Bousquet, Olivier, Chapelle, Olivier, and Hein, Matthias. Measure based regularization. In Thrun, Sebastian, Saul, Lawrence, and Schölkopf, Bernhard (eds.), *Advances in Neural Information Processing Systems (NIPS - 16)*. MIT Press, 2004. 5, 8

Davis, Steven B. and Mermelstein, Paul. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28 (4), 1980. 1

Eilers, Paul H. C. and Marx, Brian D. Flexible smoothing with B-splines and penalties. *Statistical Science*, 11(2), 1996. 1

Györfi, László, Kohler, Michael, Krzyżak, Adam, and Walk, Harro. *A Distribution-Free Theory of Nonparametric Regression*. Springer Verlag, New York, 2002. 4

Hein, Matthias. Uniform convergence of adaptive graph-based regularization. In *Proceedings of the 19th annual conference on Learning Theory (COLT)*, pp. 50–64, Berlin, Heidelberg, 2006. Springer-Verlag. 5, 6

Kim, Kwang In, Steinke, Florian, and Hein, Matthias. Semi-supervised regression using Hessian energy with an application to semi-supervised dimensionality reduction. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. K. I., and Culotta, A. (eds.), *Advances in Neural Information Processing Systems (NIPS - 22)*, pp. 979–987. 2009. 2, 8

Lowe, David G. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE International Conference on Computer Vision (ICCV)*, 1999. 1

Nadler, Boaz, Srebro, Nathan, and Zhou, Xueyuan. Semi-supervised learning with the graph laplacian: The limit of infinite unlabelled data. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. K. I., and Culotta, A. (eds.), *Advances in Neural Information Processing Systems (NIPS - 22)*, pp. 1330–1338, 2009. 8

Pearce, N. D. and Wand, M. P. Penalized splines and reproducing kernel methods. *The American Statistician*, 60(3), 2006. 1

Rifai, Salah, Mesnil, Gregoire, Vincent, Pascal, Muller, Xavier, Bengio, Yoshua, Dauphin, Yann, and Glorot, Xavier. Higher-order contractive auto-encoder. In *European Conference on Machine Learning (ECML) and Principles and Practice of Knowledge Discovery in Databases (PKDD)*, 2011. 2

Schölkopf, Bernhard and Smola, Alexander J. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002. 1

Sridharan, Karthik, Srebro, Nathan, and Shalev-Shwartz, Shie. Fast rates for regularized objectives. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L. (eds.), *Advances in Neural Information Processing Systems (NIPS - 21)*, 2009. 4

Steinwart, Ingo and Christmann, Andreas. *Support Vector Machines*. Springer, 2008. 4

Wahba, Grace. *Spline Models for Observational Data*. SIAM [Society for Industrial and Applied Mathematics], 1990. 1, 2

Wood, Simon N. Thin plate regression splines. *Journal of the Royal Statistical Society, Series B*, 65, 2003. 1

Yuille, Alan L. and Grzywacz, Norberto M. The motion coherence theory. In *Proceedings of the International Conference on Computer Vision*, 1988. 1

Zhou, Xueyuan and Belkin, Mikhail. Semi-supervised learning by higher order regularization. In *International Conference on Artificial Intelligence and Statistics*, pp. 892–900, 2011. 8

Zhu, Xiaojin, Ghahramani, Zoubin, and Lafferty, John. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the 12th International Conference on Machine LearningProceedings of the 27th International Conference on Machine Learning (ICML)*, volume 3, pp. 912–919, 2003. 8