Diffeomorphic Learning

Laurent Younes

June 2019

Introduction

- Neural networks have shown how large data transformations in high dimensions could be estimated with sufficient generalization reliability while allowing for linear class separation in the transformed space.
- Modeling these transformations is achieved by carefully designing network architectures and relying on implicit algorithmic biases to achieve generalization.
- The transformations themselves are difficult to characterize.
- We propose to estimate such transformations within a well understood class (diffeomorphisms), with explicit regularity controls.
- The approach borrows from algorithms in shape analysis, where diffeomorphic registration has become a major tool.

Prior Work

- There is of course a large body of work on the use of diffeomorphisms in shape analysis.
- Very few applications to machine learning:
  - Diffeomorphic dimensionality reduction was proposed by Warner and Schölkopf (2009) where the goal is to flatten the training data over a small number of dimensions while maintaining inter-point distances.
  - Diffeomorphic transformations of positive kernels were also introduced by Trouvé and Yu (2001).

Notation

- We consider predictors taking the form $x \in \mathbb{R}^d \mapsto F(\varphi(x))$, with
  - $\varphi : \mathbb{R}^d \to \mathbb{R}^d$: diffeomorphism.
  - $F : \mathbb{R}^d \to C = \{0, \ldots, c-1\}$: simple predictor, where c is the number of classes.
- F is parametrized by $\theta \in \mathbb{R}^q$. (Notation: $F(x; \theta)$.)
- Recall: a diffeomorphism is an invertible C^1 mapping from $\mathbb{R}^d$ onto $\mathbb{R}^d$ with a C^1 inverse.

Training Procedure

- Training set: $\mathcal{T}_0 = (x_1, y_1, \ldots, x_N, y_N)$ with $x_k \in \mathbb{R}^d$ and $y_k \in C$ for $k = 1, \ldots, N$.
- Assumption: $x_i \neq x_j$ whenever $i \neq j$.
- Notation: $\varphi \cdot \mathcal{T}_0 = (\varphi(x_1), y_1, \ldots, \varphi(x_N), y_N)$ (φ diffeomorphism of $\mathbb{R}^d$).
- Objective function: $G(\varphi, \theta) = D(\mathrm{id}, \varphi)^2 + \lambda \Gamma(F(\cdot, \theta), \varphi \cdot \mathcal{T}_0)$
- Here D is a Riemannian distance in a group of diffeomorphisms of $\mathbb{R}^d$ and Γ is a standard loss function, e.g.,

$$\Gamma(F(\cdot, \theta), \varphi \cdot \mathcal{T}_0) = -\sum_{k=1}^{N} \log F(\varphi(x_k); \theta)(y_k)$$

where $F(z, \theta)(y) = e^{\theta(y)^T z} / \sum_{y' \in C} e^{\theta(y')^T z}$ and $\theta = (\theta(1), \ldots, \theta(c-1)) \in (\mathbb{R}^d)^{c-1}$ with $\theta(0) = 0$.

Distance on Diffeomorphisms

- Let $\mathbf{B}_p = C_0^p(\mathbb{R}^d, \mathbb{R}^d)$ denote the space of p-times continuously differentiable functions that tend to 0 (together with their first p derivatives) at infinity.
- This is a Banach space, for the norm

$$\|v\|_{p,\infty} = \max_{0 \le k \le p} \|d^k v\|_\infty$$

  where $\|\cdot\|_\infty$ is the usual supremum norm.
- Introduce a Hilbert space V of vector fields on $\mathbb{R}^d$, continuously embedded in $\mathbf{B}_p$ for some $p \ge 1$, so that

$$\|v\|_{p,\infty} \le C\|v\|_V$$

  for all v in V.
- $\|\cdot\|_V$ denotes the Hilbert norm on V and $\langle \cdot, \cdot \rangle_V$ the inner product.

- This assumption implies that V is a reproducing kernel Hilbert space (RKHS).
- V is a space of vector fields: its kernel is matrix-valued. It is a function

$$K : \mathbb{R}^d \times \mathbb{R}^d \to \mathcal{M}_d(\mathbb{R})$$

such that

1. $K(\cdot, y)a : x \mapsto K(x, y)a$ belongs to V for all $y, a \in \mathbb{R}^d$.
2. For $v \in V$, for all $y, a \in \mathbb{R}^d$, $\left\langle K(\cdot, y)a\,,\, v \right\rangle_V = a^T v(y)$.

- These properties imply that $\left\langle K(\cdot, x)a\,,\, K(\cdot, y)b \right\rangle_V = a^T K(x, y)b$ for all $x, y, a, b \in \mathbb{R}^d$. (In particular $K(y, x) = K(x, y)^T$.)

- Diffeomorphisms are generated as flows of ordinary differential equations (ODEs) associated with time-dependent elements of V.
- More precisely, let $v \in L^2([0,1], V)$ if and only if $v(t) \in V$ for $t \in [0,1]$ and

$$\int_0^1 \|v(t)\|_V^2 \, dt < \infty.$$

- Then, the ODE $\partial_t y = v(t, y)$ has a unique solution over $[0,1]$ given any initial condition $y(0) = x$. The flow associated with this ODE is the function $\varphi^v : (t, x) \mapsto y(t)$ where y is the solution starting at x.
- This flow is, at all times, a diffeomorphism of $\mathbb{R}^d$, and satisfies the equation $\partial_t \varphi = v(t) \circ \varphi$, $\varphi(0) = \mathrm{id}$ (the identity map).

- The set of diffeomorphisms that can be generated in such a way form a group, denoted Diff_V since it depends on V.
- Given $\varphi_1 \in \mathrm{Diff}_V$, one defines the optimal deformation cost $\Lambda(\varphi_1)$ from id to φ_1 as the minimum of $\int_0^1 \|v(t)\|_V^2 \, dt$ over all $v \in L^2([0,1], V)$ such that $\varphi^v(1) = \varphi_1$.
- If we let $D(\varphi_1, \varphi_2) = \Lambda(\varphi_2 \circ \varphi_1^{-1})^{1/2}$, then D is a geodesic distance on Diff_V associated with the right-invariant Riemannian metric associated to $v \mapsto \|v\|_V$ on V.

- The objective function for our learning problem can be rewritten as

$$E'(v,\theta) = \int_0^1 \|v(t)\|_V^2 \, dt + \lambda \Gamma(F(\cdot,\theta), \varphi(1)\cdot \mathcal{T}_0)$$

  to be minimized over $v \in L^2([0,1],V)$, $\theta \in \mathbb{R}^q$ and subject to the constraint that $\varphi(t) = \varphi(t,\cdot)$ satisfies the equation $\partial_t \varphi = v \circ \varphi$ with $\varphi(0) = \mathrm{id}$.
- Under mild regularity conditions on the dependency of Γ with respect to $\mathcal{T}_0$ (continuity in $x_1,\ldots,x_N$ suffices), a minimizer of this function in v for fixed θ always exists, with $v \in L^2([0,1],V)$.

Reduction

- The minimization can be reduced using an RKHS argument, similar to the "kernel trick" invoked in standard kernel methods.
- Let $z_k(t) = \varphi(t, x_k)$. Because the endpoint cost Γ only depends on $(z_1(1), \ldots, z_N(1))$, it only suffices to compute these trajectories, which satisfy $\partial_t z_k = v(t, z_k)$.
- An optimal v must then take the form

$$v(t, \cdot) = \sum_{k=1}^{N} K(\cdot, z_k(t)) a_k(t)$$

where $a_1, \ldots, a_N$ are unknown time-dependent vectors in $\mathbb{R}^d$, and provide our reduced variables.

- Letting $\boldsymbol{a} = (a_1, \ldots, a_N)$, the reduced problem requires to minimize

$$E(\boldsymbol{a}(\cdot), \theta) = \int_0^1 \sum_{k,l=1}^{N} a_k(t)^T K(z_k(t), z_l(t)) a_l(t)\, dt + \lambda \Gamma(F(\cdot, \theta), \mathcal{T}(1))$$

  subject to $\partial_t z_k = \sum_{l=1}^{N} K(z_k, z_l) a_l$, $z_k(0) = x_k$, with the notation $\mathcal{T}(t) = (z_1(t), y_1, \ldots, z_N(t), y_N)$.
- This is an optimal control problem.

Gradient

- Consider the minimization of the objective function for fixed θ.
- The optimality conditions for $\boldsymbol{a}$ are provided by Pontryagin's maximum principle (PMP).
- They require the introduction of a third variable (co-state), denoted $\boldsymbol{p} \in Q = (\mathbb{R}^d)^N$, and of a control-dependent Hamiltonian $H_{\boldsymbol{a}}$ defined on $Q \times Q$ given, in our case, by

$$H_{\boldsymbol{a}}(\boldsymbol{p}, \boldsymbol{z}) = \sum_{k,l=1}^{N} (p_k - a_k)^T K(z_k, z_l) a_l.$$

The PMP then states that any optimal solution $\boldsymbol{a}$ must be such that there exists a time-dependent co-state satisfying

$$\begin{cases} \partial_t \boldsymbol{z} = \partial_{\boldsymbol{p}} H_{\boldsymbol{a}}(t)(\boldsymbol{p}(t), \boldsymbol{z}(t)) \\ \partial_t \boldsymbol{p} = -\partial_{\boldsymbol{z}} H_{\boldsymbol{a}}(t)(\boldsymbol{p}(t), \boldsymbol{z}(t)) \\ \boldsymbol{a}(t) = \mathrm{argmax}_{\boldsymbol{a}'} H_{\boldsymbol{a}'}(\boldsymbol{p}(t), \boldsymbol{z}(t)) \end{cases}$$

with boundary conditions $\boldsymbol{z}(0) = (x_1, \ldots, x_N)$ and

$$\boldsymbol{p}(1) = -\lambda \partial_{\boldsymbol{z}} \Gamma(F(\cdot, \theta), \mathcal{T}(1)).$$

. . .

The differential of E with respect to $\boldsymbol{a}(\cdot)$ is given by

$$\partial_{\boldsymbol{a}(\cdot)} E(\boldsymbol{a}(\cdot), \theta) = \boldsymbol{u}(\cdot)$$

with

$$u_k(t) = \sum_{l=1}^{N} K(z_k(t), z_l(t))(p_l(t) - 2a_l(t))$$

where p solves

$$\begin{cases} \partial_t \boldsymbol{z} = \partial_{\boldsymbol{p}} H_{\boldsymbol{a}}(t)(\boldsymbol{p}(t), \boldsymbol{z}(t)) \\ \partial_t \boldsymbol{p} = -\partial_{\boldsymbol{z}} H_{\boldsymbol{a}}(t)(\boldsymbol{p}(t), \boldsymbol{z}(t)) \end{cases}$$

with boundary conditions $\boldsymbol{z}(0) = (x_1, \ldots, x_N)$ and

$$\boldsymbol{p}(1) = -\lambda \partial_{\boldsymbol{z}} \Gamma(F(\cdot, \theta), \mathcal{T}(1)).$$

Discrete Version

- Discretize time over $0, 1, \ldots, T$ and minimize the objective function

$$E(\boldsymbol{a}(\cdot), \theta) = \frac{1}{T} \sum_{t=0}^{T-1} \sum_{k,l=1}^{N} a_k(t)^T K(z_k(t), z_l(t)) a_l(t)\, dt + \lambda \Gamma(F(\cdot, \theta), \mathcal{T}(T))$$

  subject to $z_k(t+1) = z_k(t) + \frac{1}{T} \sum_{l=1}^{N} K(z_k(t), z_l(t)) a_l(t)$, $z_k(0) = x_k$.

- Then $\partial_{\boldsymbol{a}(\cdot)} E(\boldsymbol{a}(\cdot), \theta) = \boldsymbol{u}(\cdot)$, with $u_k(t) = \sum_{l=1}^{N} K(z_k(t), z_l(t))(p_l(t) - 2a_l(t))$, $t = 0, \ldots, T-1$, where p (discretized over $0, \ldots, T-1$), can be computed using

$$\begin{cases} \boldsymbol{z}(t+1) = \boldsymbol{z}(t) + \dfrac{1}{T} \partial_{\boldsymbol{p}} H_{\boldsymbol{a}}(t)(\boldsymbol{p}(t), \boldsymbol{z}(t)) \\ \boldsymbol{p}(t-1) = \boldsymbol{p}(t) + \dfrac{1}{T} \partial_{\boldsymbol{z}} H_{\boldsymbol{a}}(t)(\boldsymbol{p}(t), \boldsymbol{z}(t)) \end{cases}$$

  with boundary conditions $\boldsymbol{z}(0) = (x_1, \ldots, x_N)$ and $\boldsymbol{p}(T-1) = -\lambda \partial_{\boldsymbol{z}} \Gamma(F(\cdot, \theta), \mathcal{T}(T))$.

Kernels

- To fully specify the method, one needs to choose the RKHS V, or, equivalently, its reproducing kernel, K.
- K specifies the regularity of the estimated diffeomorphism.
- Recall that K is matrix valued.
- Simple example: "Scalar" kernel. Let $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a positive kernel and let

$$K(x, y) = \kappa(x, y)\mathrm{Id}_{\mathbb{R}^d}.$$

- Gaussian kernel:

$$\kappa(x, y) = \exp(-|x - y|^2/2a^2)$$

- Matérn kernels:

$$\kappa(x, y) = P_k(|x - y|/a) \exp(-|x - y|/a)$$

where P_k is a reversed Bessel polynomial of order k.

- Vector fields v in the RKHS associated with $K = \kappa \mathrm{Id}$, where κ is a radial basis function (RBF), are such that each coordinate function of v belongs to the scalar RKHS associated with κ.
- This space is translation and rotation invariant.
- In particular, the norm on V is invariant after permutation of the indices.

Graph-based Kernels

- Assume that the input data is supported by a graph $\mathcal{G}$ with d vertices.
- Let $\mathcal{N}_i$ denote the set of nearest neighbors of i in $\mathcal{G}$.
- Define
$$K(x,y) = \mathrm{diag}(\kappa(|P_i x - P_i y|), i = 1, \ldots, d)$$
where $P_i x$ is the vector $(x_j, j \in \mathcal{N}_i)$.
- In the associated RKHS, vector fields $v \in V$ are such that $v^{(i)}(x)$ (their ith coordinate) only depends on $P_i(x)$
- Extreme case: $P_i(x) = x_i$ (naive kernel).
- Such kernels are not invariant by the full rotation group of $\mathbb{R}^d$.

Adding Affine Motion

- Vector fields in the RKHS V vanish at infinity, and the resulting diffeomorphic flow satisfy $\varphi(x) \simeq x$ for large x.
- One can include an affine component in the motion, adding new control variables $A(t)$ and $B(t)$ completing the previous ones so that

$$\partial_t z_k = A z_k + B + \sum_{l=1}^{N} K(z_k, z_l) a_l$$

- One also needs to add a penalty, e.g., $\mu \int_0^1 \|A(t)\|^2 dt + \mu \int_0^1 \|B(t)\|^2 dt$ to the objective function.
- The modifications to the optimality conditions (PMP) and to the gradient computation are straightforward.

Remark

- This is not "deformable templates."
- A deformable template algorithm typically works with small dimensional images (k=2 or 3), say $I : \mathbb{R}^k \to \mathbb{R}$, and transforms it as $I \circ g^{-1}$ (using a diffeomorphism $g : \mathbb{R}^k \to \mathbb{R}^k$) to compare it with a target (e.g., class average).
- The transformation $\varphi_g : I \mapsto I \circ g^{-1}$ is a homeomorphism of the space of, say, continuous images.
- Once images are discretized over a grid with d points, φ_g becomes a one-to-one transformation of $\mathbb{R}^d$, but a very special one.
- Our model optimizes over all possible transformations of $\mathbb{R}^d$!

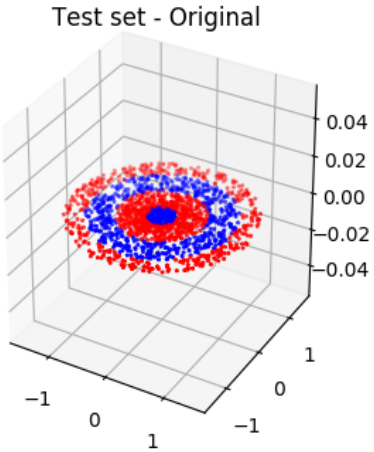
Adding a dimension

- We use logistic regression as final classifier applied to transformed data.
- Obviously, training sets cannot always by transformed into a linearly separable dataset using a diffeomorphism.
- Example (in 1D): associate class 0 to $x \in \mathbb{R}$ is 0 if $|x| < 1$ and class 1 otherwise.
- Simple solution: add a dummy dimension to the data.
- With this choice, any binary classifier $x \mapsto \mathrm{sign}(f(x) - a)$ for some smooth function f can be included in the model class: letting μ denote the additional scalar variable, defining $\varphi(x, \mu) = (x, \mu + f(x))$ makes the data linearly separable.
- This may not be the optimal solution, since the classifier will trade off some non-linear transformation of the data (x) in order to induce a "simpler" classification rule.

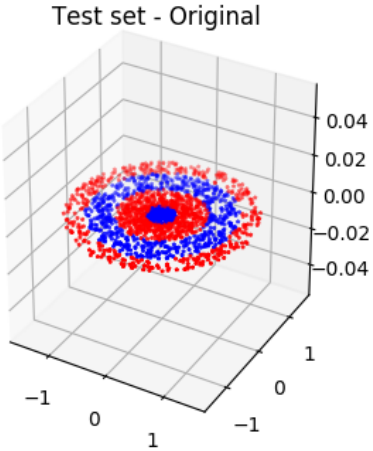
Implementation

- The final classifier minimizes
$$-\sum_{k=1}^{N} \log F(\varphi(x_k);\theta)(y_k) + \lambda \sum_{i=1}^{c-1}\sum_{j=1}^{d} \theta_{ij}^2 \sigma_j^2$$
where σ_j is the standard deviation of the jth coordinate of $(\varphi(x_1),\ldots,\varphi(x_N))$ and F is the logistic likelihood.
- Time was discretized in 10 intervals.
- We used L-BFGS to minimize the objective function with respect to all parameters together.
- Before learning, the data is rescaled so that the median distance between two training x's is 1, and a fixed value for the kernel width parameter (e.g., 1.0) is selected.
- The penalty coefficient λ in front of the cost function Γ is progressively increased during learning until a target value of the training error is reached (0.005 in our experiments).
- The procedure is stopped if the gradient is small or if the objective function's variation falls below a threshold.

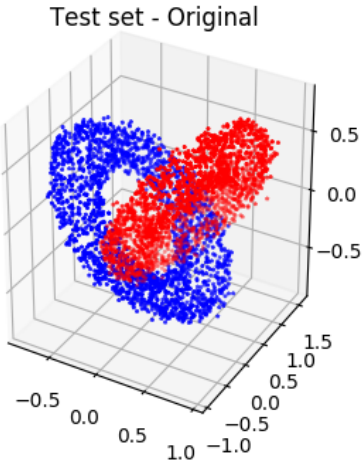
2D Example: "Target"



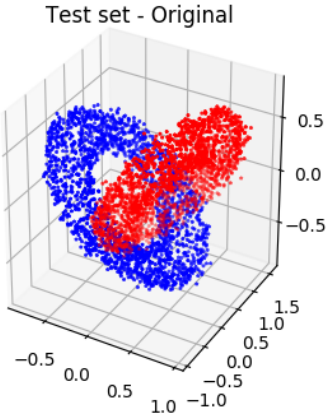
2D Example: "Target"



2D Example: Tori



2D Example: Tori



Some comparative runs

- We tried this model on small- to average-dimensional classification problems, on relatively small datasets.
- We compared the results before and after transformation with standard classifiers using the scikit-learn package.

	Log. Reg.	linSVM	SVM	RF	kNN	MLP1	MLP2	MLP5
"Target" (500 training examples)								
Original Data	0.529	0.527	0.436	0.134	0.159	0.392	0.082	0.070
Transformed Data	0.047	0.045	0.047	0.0472	0.050	0.048	0.045	0.045
3D tori (200 training examples)								
Original Data	0.334	0.338	0.053	0.028	0.004	0.057	0.002	0.001
Transformed Data	0.000	0.000	0.000	0.000	0.000	0.004	0.001	0.000

Classification Error Rates

Classifiers (using scikit learn)

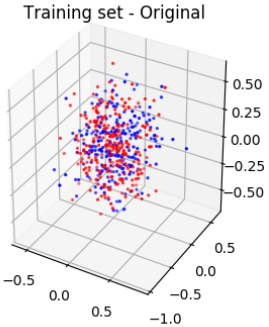
- Log. Reg.: (Ridge) Logistic Regression (part of the diff. learning optimization problem).
- Lin. SVM: Linear SVM (ℓ^2 penalty, squared-hinge loss, $C = 1$.)
- SVM: RBF with $\gamma = 1$ (data was normalized) and $C = 1$.
- RF: Random forest, with 20 trees.
- kNN: k-nearest neighbors ($k = 5$, Euclidean metric).
- MLP1, MLP2, MLP5: multi-layer perceptrons with 1, 2, 5 hidden layers, 100 units per layer (using ReLU activations, SGD–ADAM, 10,000 maximum iterations).

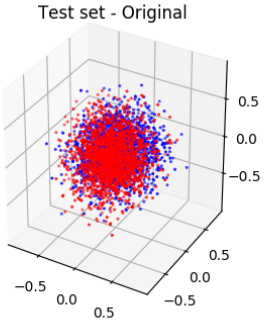
Tori

Same tori configuration in 3D, now immersed in a larger-dimensional space and followed by a random rotation.

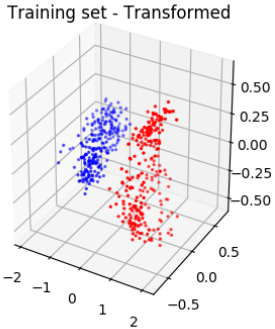
	Log. Reg.	linSVM	SVM	RF	kNN	MLP1	MLP2	MLP5
	Tori in 10 dimensions, 200 training samples							
Original Data	0.324	0.330	0.296	0.315	0.306	0.170	0.171	0.224
Transformed Data	0.207	0.208	0.206	0.209	0.203	0.210	0.204	0.209
	Tori in 10 dimensions, 500 training samples							
Original Data	0.325	0.328	0.260	0.284	0.249	0.071	0.071	0.103
Transformed Data	0.073	0.070	0.070	0.073	0.066	0.076	0.077	0.085
	Tori in 20 dimensions, 200 training samples							
Original Data	0.321	0.324	0.312	0.385	0.392	0.330	0.330	0.337
Transformed Data	0.324	0.324	0.325	0.323	0.324	0.315	0.318	0.321
	Tori in 20 dimensions, 500 training samples							
Original Data	0.317	0.321	0.298	0.336	0.337	0.179	0.238	0.298
Transformed Data	0.281	0.282	0.279	0.285	0.283	0.279	0.279	0.281

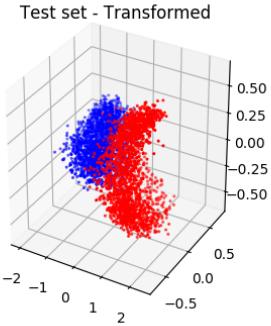
Before and After Training: Tori





Before and After Training: Tori



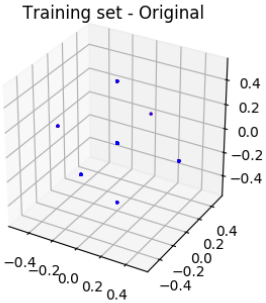


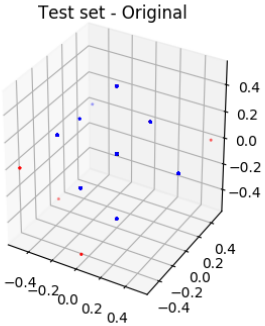
Xor

50-dimensional data with all zero coordinates except two (chosent at random) equal to ± 1. Classes result from exclusive or applied to these two coordinates.

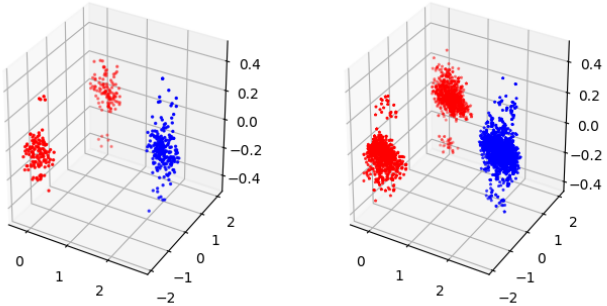
	Log. Reg.	linSVM	SVM	RF	kNN	MLP1	MLP2	MLP5
		xor, 200 training samples						
Original Data	0.502	0.504	0.491	0.479	0.482	0.215	0.381	0.428
Transformed Data	0.043	0.044	0.028	0.052	0.030	0.022	0.024	0.024

Before and After Training: XOR





Before and After Training: XOR



Thresholded sum of RBF

- Classes are given by the signs of

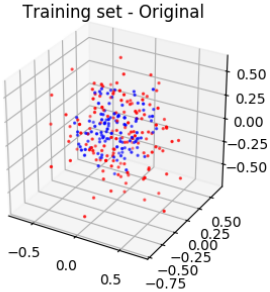
$$\sin\left(\sum_{j=1}^{L} \rho(|X - c_j|) a_j\right) - \mu \text{ with } \rho(z) = \exp(-z^2)$$

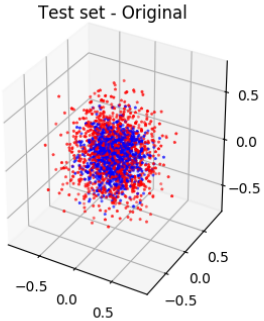
  where X is a d-dimensional standard Gaussian and μ is estimated so that both positive and negative classes are balanced.
- The centers are $c_j = (j/2L) e_{(j \bmod d)+1}$ where $j \bmod d$ is the remainder of the division of j by d and $e_1, \ldots, e_d$ is the canonical basis of $\mathbb{R}^d$.
- The coefficients are $a_j = 2(j \bmod 1.5) - 0.75$.

Thresholded sum of RBF (Results)

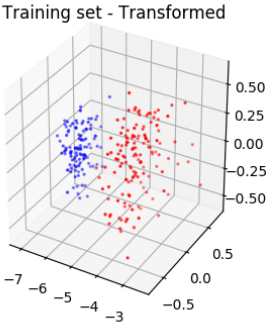
	Log. Reg.	linSVM	SVM	RF	kNN	MLP1	MLP2	MLP5
			RBF, 200 training samples					
Original Data	0.521	0.517	0.157	0.324	0.382	0.297	0.233	0.206
Transformed Data	0.145	0.146	0.156	0.172	0.141	0.161	0.163	0.159
			RBF, 500 training samples					
Original Data	0.500	0.505	0.103	0.208	0.390	0.209	0.205	0.130
Transformed Data	0.142	0.139	0.154	0.146	0.139	0.157	0.152	0.157

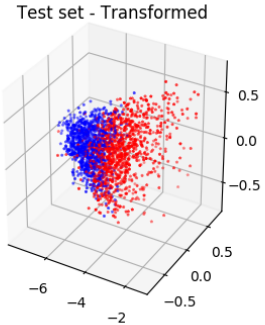
Before and After Training: RBF





Before and After Training: RBF





Discussion

- These are preliminary results that however show promising properties of diffeomorphic learning.
- More work is needed to make the approach computationally feasible for large scale problems.
- (There is, for example, no gain when using a basic version of SGD.)
- Sub-Riemannian approaches for the definition of the distance may reduce the complexity and should be a direction to explore. (They would reduce the space of vector fields, including affine transformations, that are allowed at each time step.)
- Extension to other types of machine learning problems need to be developed, too. (Regression is straightforward.)

Regression for the 2D "target" example