eman ta zabal zazu

Towards Relevant Dendritic Computing

Manuel Graña, Ana I. Gonzalez-Acuña,
Computational Intelligence Group, UPV/EHU

outline

- Motivation
- Lattice kernels
- Learning lattice kernels
- Sparse Bayesian Learning
- SBL for lattice kernels: relevant dendritic computing
- Experimental Results
- Conclusions

Motivation

- Dendritic computing (DC):
  - Single Layer Morphological Neuron

$$\tau_j(\mathrm{x}) = p_j \bigwedge_{i \in I_j} \bigwedge_{l \in L_{ij}} (-1)^{1-l} \left(x_i + w_{ij}^l\right),$$

$$\tau(\mathrm{x}) = \bigwedge_{k=1}^{j} \tau_k(\mathrm{x});\ \xi = 1, \ldots, m.$$

Motivation

- Problems of DC
  - Ad-hoc learning algorithm
  - Overfitting
  - Lack of regularization
- Aims of the work
  - Improve generalization
  - Propose a general learning framework
  - Introduce regularization to obtain sparseness

Strategy

- Define a variation of SLMN

$$\tau_j(\mathrm{x}) = p_j \bigwedge_{i \in I_j} \left[\pi_{ij}^l + (x_i - l_{ij}) \wedge \pi_{ij}^u + (u_{ij} - x_i) \right],$$

- Which can be assimilated to “lattice kernels”
- Apply Sparse Bayesian Learning to obtain
  - sparseness
  - generalization

Lattice kernels

- Lattice kernel formulation of the SLMN
  - System's response

$$\tau(\mathbf{x}) = \bigvee_{n=1}^{N} \left(\pi_n + \lambda_n(\mathbf{x}, \mathbf{x}_n) p_n\right), \quad p_n \in \{-1, 1\}$$

  - Kernel response

$$\lambda_n(\mathbf{x}, \mathbf{x}_n) = \bigwedge_{i=1}^{d} \left[\pi_{ni}^{l} + \left(x_i - \left(x_{ni} - \varepsilon_{ni}^{l}\right)\right) \wedge \pi_{ni}^{u} + \left(\left(x_{ni} + \varepsilon_{ni}^{u}\right) - x_i\right)\right],$$

$$\pi_{ni}^{l}, \pi_{ni}^{u} \in \{0, \infty\} \qquad \varepsilon_{ni}^{l}, \varepsilon_{ni}^{u} \in \mathbb{R}^{+}.$$

Learning Lattice kernels

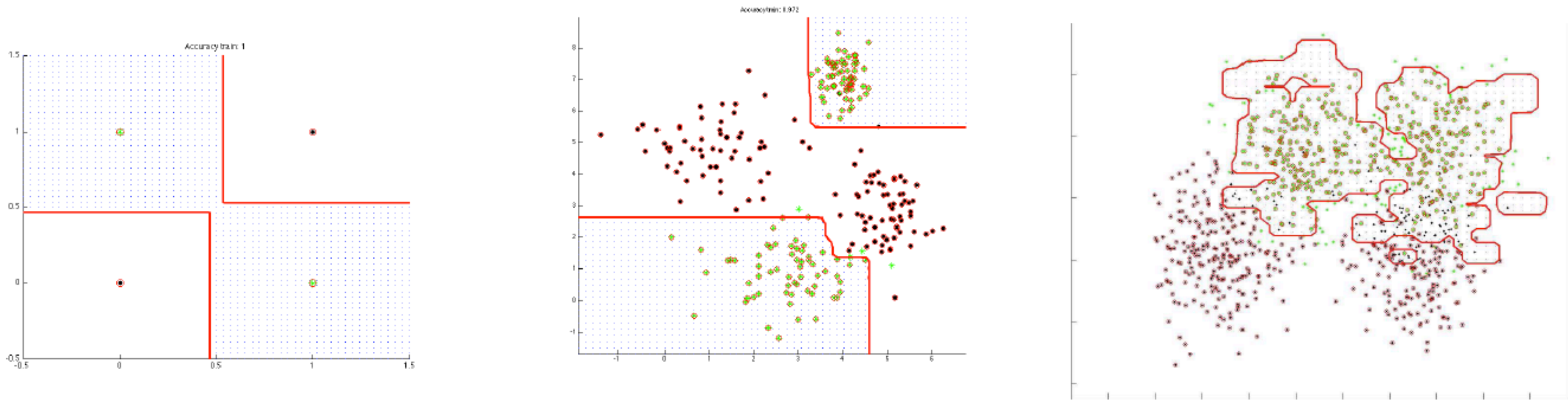
- **Algorithm 2** Monte Carlo method for the training of the SLKN

  Initialize randomly $\pi(0)$, compute $E(0)$
  Set the initial temperature $T(0)$
  $k = 0$
  Repeat

  - generate a random candidate configuration $\pi'(k)$
  - compute $E'(k)$
  - $\triangle E = E'(k) - E(k)$
  - compute $P_a(\triangle E, T) = e^{\triangle E/T}$, generate random $r \sim \mathcal{U}(0,1)$
  - if $\triangle E > 0$ or $P_a(\triangle E, T) > r$ then $\pi(k+1) = \pi'(k)$; $E(k+1) = E(k)$.
  - reduce T

  Until convergence

Learning lattice kernels



(a) (b) (c)

Figure 1: Distribution of class 1 (blue dot region) obtained by training on the (a) XOR, (b) Gaussians centered at the XOR points, (c) the synthetic data used by Tipping [20].

Sparse Bayesian Learning

- Data likelihood 2-class classification

$$P(\mathbf{t}\,|\mathbf{w}) = \prod_{n=1}^{N} f\left(y\left(\mathbf{x}_n;\mathbf{w}\right)\right)^{t_n}\left[1 - f\left(y\left(\mathbf{x}_n;\mathbf{w}\right)\right)\right]^{1-t_n}.$$

- A priori of the weights

$$p\left(\mathbf{w}\,|\boldsymbol{\alpha}\right) = \prod_{i=0}^{N} \mathcal{N}\left(w_i\,\middle|0, \alpha_i^{-1}\right),$$

Sparse Bayesian Learning

- Hyperparameters "uninformative" a priori

$$p(\boldsymbol{\alpha}) = \prod_{i=0}^{N} \mathrm{Gamma}(\alpha_i | a, b),$$

- Goal: simultaneous estimation of weights and hyperparameters

$$p(\mathbf{w}, \boldsymbol{\alpha} | \mathbf{t}) = p(\mathbf{w} | \mathbf{t}, \boldsymbol{\alpha})\, p(\boldsymbol{\alpha} | \mathbf{t}),$$

Sparse Bayesian Learning

- Decomposed into the posterior of the weights

$$p(\mathbf{w}\,|\mathbf{t},\boldsymbol{\alpha}) = \frac{p(\mathbf{t}\,|\mathbf{w})\,p(\mathbf{w}\,|\boldsymbol{\alpha})}{p(\mathbf{t}\,|\boldsymbol{\alpha})} \propto p(\mathbf{t}\,|\mathbf{w})\,p(\mathbf{w}\,|\boldsymbol{\alpha}).$$

- Posterior of the hyperparameters

$$p(\boldsymbol{\alpha}\,|\mathbf{t}) \propto p(\mathbf{t}\,|\alpha)\,p(\boldsymbol{\alpha}),$$

- Uniformative prior implies that is enough to compute $p(\mathbf{t}\,|\alpha)$

Relevant dendritic computing

- Data likelihood based on the lattice kernels model

$$P(\mathbf{t}\,|\boldsymbol{\pi},\boldsymbol{\varepsilon}) = \prod_{n=1}^{N} f\left(\tau\left(\mathbf{x}_n;\boldsymbol{\pi},\boldsymbol{\varepsilon}\right)\right)^{t_n}\left[1 - f\left(\tau\left(\mathbf{x}_n;\boldsymbol{\pi},\boldsymbol{\varepsilon}\right)\right)\right]^{1-t_n},$$

- Spike and slab prior of the parameters

$$p(\boldsymbol{\varepsilon}\,|C,\boldsymbol{\alpha}) = \prod_{n=1}^{N}\prod_{i=1}^{d}\prod_{k\in\{l,u\}}\left[(1-C)\,\delta\left(\left(\varepsilon_{ni}^{k}\right)^{-1}\right) + C\mathcal{E}\left(\varepsilon_{ni}^{k}\,\middle|\left(\alpha_{ni}^{k}\right)^{-1}\right)\right],$$

Relevant dendritic computing

- It is possible to formulate the log-posterior of the weights

$$\begin{aligned}\log p\left(\boldsymbol{\varepsilon}\left|\mathbf{t}, C, \boldsymbol{\alpha}\right.\right) \quad = \quad & \sum_{n=1}^{N}\left[t_n \log y_n + (1-t_n)\log\left[1-y_n\right]\right] \\ & - \sum_{\left\{n,i,k\left|\pi_{ni}^{k}=0\right.\right\}} \left[\log\left(\alpha_{ni}^{k}\right) + \varepsilon_{ni}^{k}\left(\alpha_{ni}^{k}\right)^{-1}\right] \\ & + \sum_{\{n,i,k\}} \left[\delta\left(\pi_{ni}^{k}\right)\log C + \left(1-\delta\left(\pi_{ni}^{k}\right)\right)\log\left(1-C\right)\right]\end{aligned}$$

Relevant dendritic computing

Algorithm 3 The Sparse Bayesian Learning applied to the SLKN

1. Initialize hyperparameters at uninformative values $\alpha_{ni}^{k} = Nd$, $C = 0.5$
2. Search for the most probable weights $(\boldsymbol{\varepsilon}, \boldsymbol{\pi})_{MP}$ maximizing by Monte-Carlo Methods the log-posterior of equation (20)

$$\boldsymbol{\varepsilon}_{MP} = \arg\max_{(\boldsymbol{\varepsilon},\boldsymbol{\pi})} \log p\left(\boldsymbol{\varepsilon} \left| \mathbf{t}, \boldsymbol{\pi}, C, \boldsymbol{\alpha} \right.\right)$$

   where $y_n = f\left(\tau\left(\mathbf{x}_n; \boldsymbol{\pi}, \boldsymbol{\varepsilon}\right)\right)$.

3. Update the hyperparameters $\alpha_{ni}^{k,\mathrm{new}} = \left(\varepsilon_{ni}^{k}\right)_{MP}^{-1}$.
4. Set relevant parameters: set $\pi_{ni}^{k} = \infty$ if $\alpha_{ni}^{k} < \epsilon$.
5. $C = 1 - \left|\left\{\alpha_{ni}^{k} < \epsilon\right\}\right| / 2Nd$
6. Test convergence. If not converged, repeat from step 2.

Experimental results

dataset	#train	#test	dimension	Best error result reported
flare solar	666	400	9	32.43 +/- 1.82
breast	200	77	9	24.77 +/- 4.63
titanic	150	2051	3	22.58 +/- 1.18
thyroid	140	75	5	4.20 +/- 2.07
heart	170	100	13	15.95 +/- 3.26
diabetes	468	300	8	23.21 +/- 1.63
german	700	300	20	23.61 +/- 2.07
synth	250	1000	2	–

Table 1: Information about the experimental classification datasets

Experimental results

dataset	RVM	SLKN	SLKN-G	SBL-SLKN
flare solar	0.65(.68)	0.56(0.55)	0.55(0.55)	0.62(0.62)
breast	0.73(0.75)	0.74(0.99)	0.73(0.99)	0.74(0.95)
titanic	0.77(0.8)	0.33(0.29)	0.33(0.29)	0.74(0.70)
thyroid	0.88(0.91)	0.92(0.99)	0.91(0.99)	0.92(0.98)
heart	0.80(0.89)	0.72(1)	0.67(1)	0.77(0.98)
diabetes	0.76(0.79)	0.74(0.94)	0.75(0.98)	0.75(0.94)
german	0.79(0.76)	0.78(1)	0.73(1)	0.78(0.98)
synth	0.90(0.87)	0.86(0.86)	0.87(0.86)	0.89(0.9)

Table 2: Classification Accuracy results on the test (train) partition

Conclusions

- Lattice kernel classification model
- Trained with Monte Carlo Methods
- Application of Sparse Bayesian Learning,
- Results on benchmark datasets are promising
- Less sparse than RVM results

- Thank you for your attention