











深度极端学习机



极端学习机与深度学习

2

- ▶ 深度学习和极端学习机被Google评为近十年来被引用最多的两篇文章。

类别 > Engineering & Computer Science > Artificial Intelligence ▾			2006
标题	作者	引用次数	
Reducing the dimensionality of data with neural networks GE Hinton , RR Salakhutdinov - science, 2006	 Geoffrey Hinton	5052	
Extreme learning machine: theory and applications GB Huang , QY Zhu , CK Siew - Neurocomputing, 2006	 Guang-Bin Huang	4389	
Markov logic networks M Richardson , P Domingos - Machine learning, 2006	 Matthew Richardson	2180	
Extremely randomized trees P Geurts , D Ernst , L Wehenkel - Machine learning, 2006	 Pierre Geurts	1235	
The DLV system for knowledge representation and reasoning N Leone , G Pfeifer , W Faber , T Eiter - ACM Transactions on ..., 2006	 Nicola Leone	1095	
A GA-based feature selection and parameters optimization for support vector machines CL Huang , GL Wang - Expert Systems with applications, 2006	 王家宏 Wang	960	
A fast and accurate online sequential learning algorithm for feedforward networks NY Liang , GB Huang , P Saratchandran - IEEE Transactions on ..., 2006	 Sundararajan N	949	
The max-min hill-climbing Bayesian network structure learning algorithm I Tsamardinos , LE Brown , CF Aliferis - Machine learning, 2006	 Ioannis Tsamardinos	865	
Polychronization: computation with spikes EM Izhikevich - Neural computation, 2006	 Eugene M. Izhikevich	781	
An integrated trust and reputation model for open multi-agent systems TD Huynh , UR Jennings - Autonomous Agents and ..., 2006	 Trung Dong Huynh	688	

计算机程序会自动估算并确定日期和引用次数。

深度极端学习机

- 问题和动机：目前的极端学习机方法及其改进的算法主要运用了单隐层的前向反馈神经网络，而没有采用比较流行和有效的层叠泛化准则去寻找输入数据的深层表征。深层模型可以找到高级别的特征表示，从而可以学习出更加抽象的特征，但是目前的深度学习方法需要解决复杂而且非凸的优化问题。
- 提出了基于极端学习机的层叠模型DrELM，根据层叠泛化准则，采用极端学习机学习深层表征。该模型采用极端学习机作为基本的层构建方法，以随机平移与核化作为层叠的基本元素。

Wenchao Yu, Fuzhen Zhuang, Qing He and Zhongzhi Shi. Learning Deep Representations via Extreme Learning Machine, Neurocomputing, Volume 149, Part A, 3 February 2015, Pages 308-315

极限学习机简介

- 极限学习机 (ELM) 可以写成 $H\beta = T$ 的形式, 其中

$$H = \begin{bmatrix} h(x_1) \\ \vdots \\ h(x_m) \end{bmatrix} = \begin{bmatrix} G(a_1, b_1, x_1) & \cdots & G(a_L, b_L, x_1) \\ \vdots & \ddots & \vdots \\ G(a_1, b_1, x_m) & \cdots & G(a_L, b_L, x_m) \end{bmatrix}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times c} \quad \text{and} \quad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_m^T \end{bmatrix}_{m \times c}$$

- H 是隐层的输出矩阵, $(x_i, t_i) \in \mathcal{R}^d \times \mathcal{R}^c$ 是训练样本集合,

$G(a, b, x)$ 是隐层特征的映射函数。

极限学习机简介

- 极限学习机 (ELM) 可以写成 $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$ 的形式, 其中

$$\mathbf{H} = \begin{bmatrix} h(\mathbf{x}_1) \\ \vdots \\ h(\mathbf{x}_m) \end{bmatrix} = \begin{bmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_m) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_m) \end{bmatrix}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times c} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} t_1^T \\ \vdots \\ t_m^T \end{bmatrix}_{m \times c}$$

- \mathbf{H} 是隐层的输出矩阵, $(\mathbf{x}_i, t_i) \in \mathcal{R}^d \times \mathcal{R}^c$ 是训练样本集合, $G(\mathbf{a}, b, \mathbf{x})$ 是隐层特征的映射函数。

Algorithm 1. Basic ELM.

Input: A training set $\mathbf{X} = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathcal{R}^d, \mathbf{t}_i \in \mathcal{R}^c, i = 1, 2, \dots, m\}$,
hidden node number L , hidden node transfer function
 $G(\cdot), j = 1, 2, \dots, L$.

Output: The prediction function of ELM for classification and regression.

(1) Randomly generate hidden node parameters matrix

$$\mathbf{W} \in \mathcal{R}^{d \times L};$$

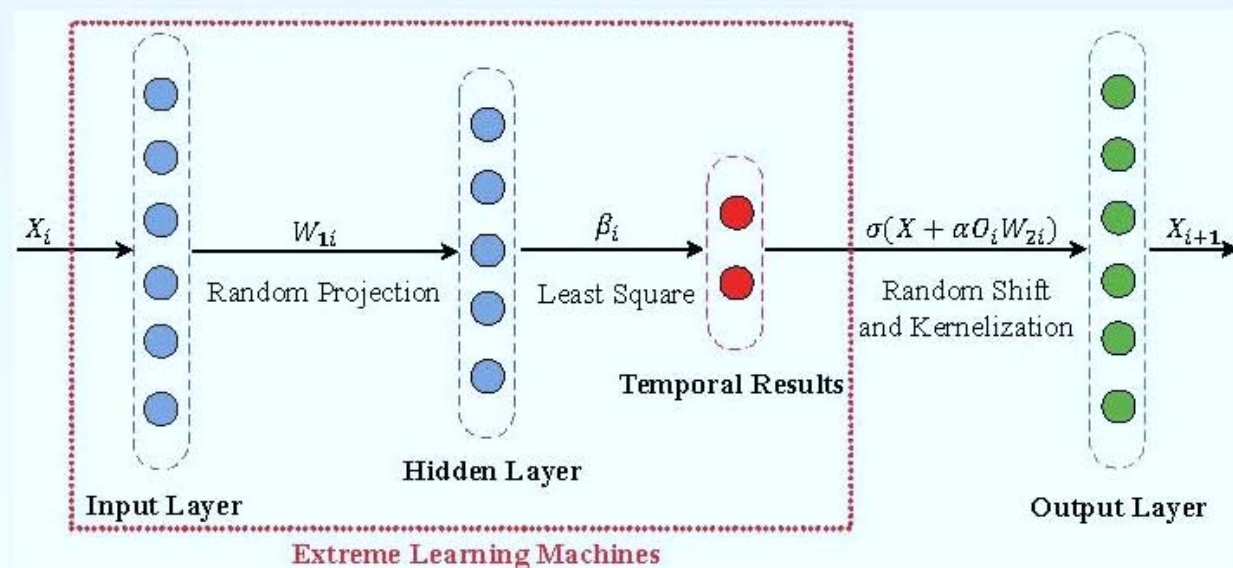
(2) Compute the hidden layer output matrix $\mathbf{H} = G(\mathbf{W}, \mathbf{X})$;

(3) Compute output weight vector $\hat{\boldsymbol{\beta}} = \mathbf{H}^\dagger \mathbf{T}$;

(4) Compute the result decision function:

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x})\hat{\boldsymbol{\beta}} = \mathbf{h}(\mathbf{x})\mathbf{H}^\dagger \mathbf{T}.$$

深度极限学习机 (DrELM)



- 首先将输入数据 X_i 随机映射到隐层空间；
- 求解最小二乘问题 $\|W_{1i}X_i\beta_i - T\|$ ，从而学习出系数 β_i ；
- 基于分类和回归结果 O_i 随机平移原始特征，并核化 $\sigma(\cdot)$ 。

DrELM 算法

- ① 选择模型的深度 $k \in \mathcal{R}$;
- ② 初始化 $\mathbf{X}_1 = \mathbf{X}$;
- ③ **for** i from 1 to k
- ④ 随机生成系数矩阵 $\mathbf{W}_{1i} \in \mathcal{R}^{d \times L}$ 和 $\mathbf{W}_{2i} \in \mathcal{R}^{c \times d}$;
- ⑤ 计算隐层输出 $\mathbf{H}_i = \mathbf{X}_i \mathbf{W}_{1i}$;
- ⑥ 计算输出权重向量 $\hat{\boldsymbol{\beta}}_i = \mathbf{H}_i^\dagger \mathbf{T}$;
- ⑦ 计算分类或回归结果 $\mathbf{O}_i = \mathbf{H}_i \hat{\boldsymbol{\beta}}_i$;
- ⑧ 计算 \mathbf{X}_{i+1} : $\mathbf{X}_{i+1} = \sigma(\mathbf{X} + \alpha \mathbf{O}_i \mathbf{W}_{2i})$;
- ⑨ **end**
- ⑩ 求解预测函数 $f_k(\mathbf{X}) = \mathbf{X}_k \mathbf{W}_{1k} \hat{\boldsymbol{\beta}}_k$.

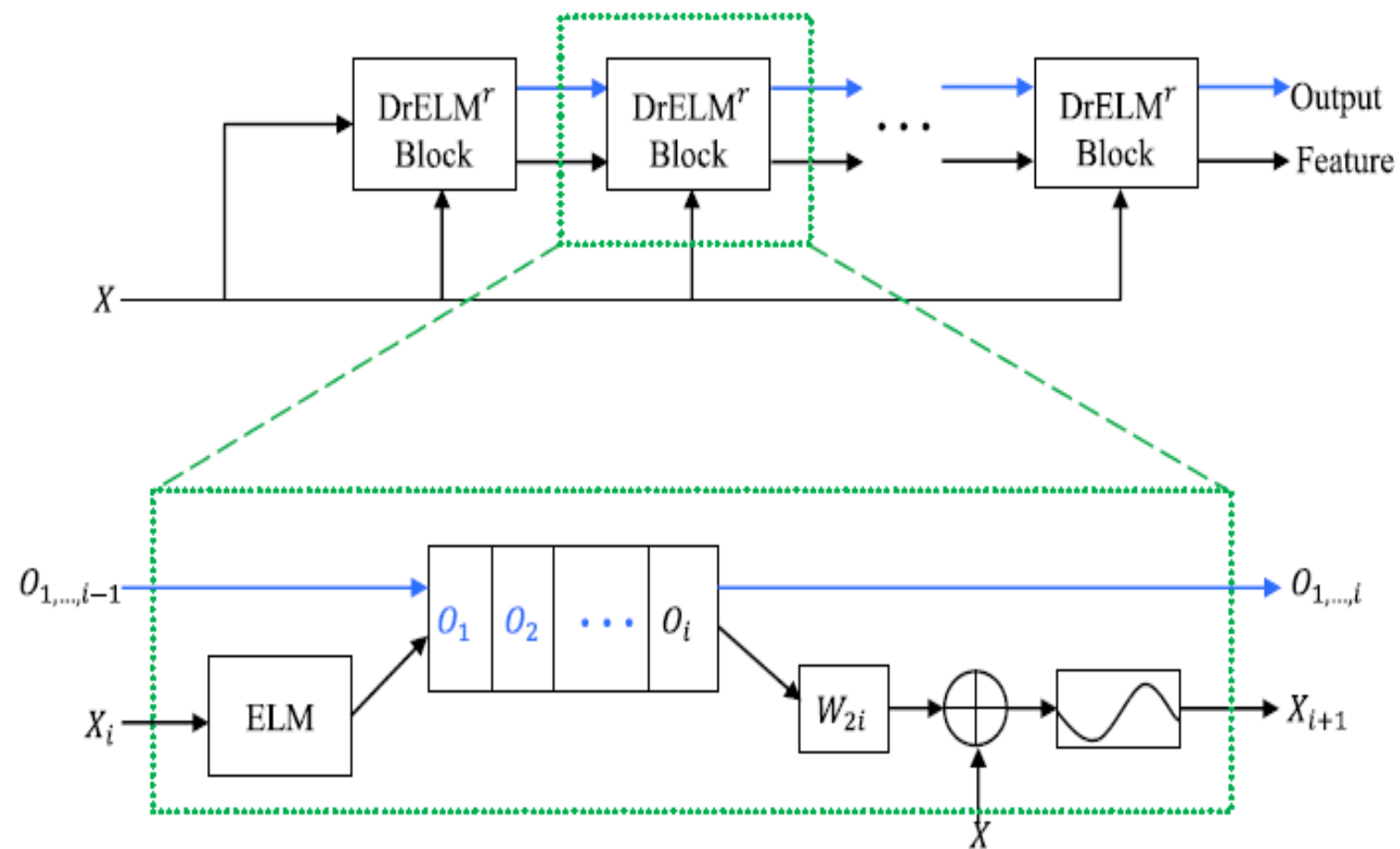


Fig. 2. Illustration of DrELM^r paradigm. In recursive version, X_{i+1} updates with all the prediction results in the previous layers O_1, O_2, \dots, O_i .

Algorithm 2. DrELM.

Input: The training set $\mathbf{X} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(m)}]^T$ and $\mathbf{T} = [\mathbf{t}^{(1)}, \mathbf{t}^{(2)}, \dots, \mathbf{t}^{(m)}]^T$ corresponding to each training instance, $\mathbf{x}^{(i)} \in \mathcal{R}^d$, $\mathbf{t}^{(i)} \in \mathcal{R}^c$. The hidden node number L .

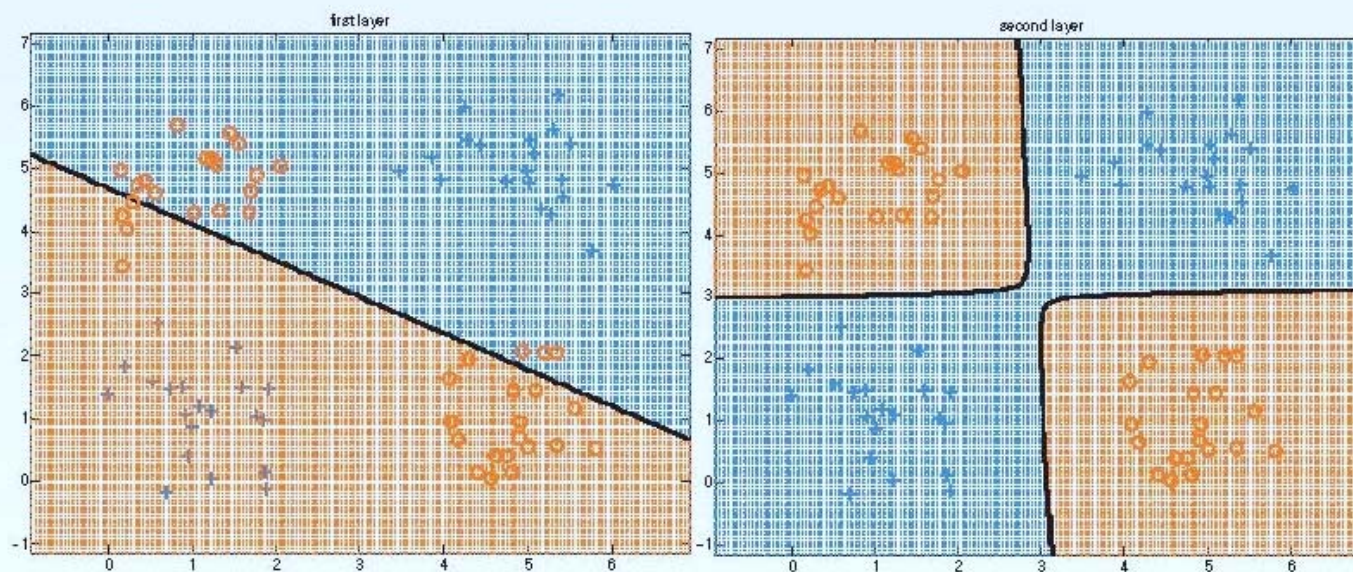
Output: The prediction function of DrELM for regression or classification.

- (1) Choose model depth k ;
- (2) Initial $\mathbf{X}_1 = \mathbf{X}$;
- (3) **for** i from 1 to k
- (4) Randomly generate hidden node parameters matrix $\mathbf{W}_{1i} \in \mathcal{R}^{d \times L}$;
- (5) Compute hidden layer output $\mathbf{H}_i = \mathbf{X}_i \mathbf{W}_{1i}$;
- (6) Compute output weight vector $\hat{\boldsymbol{\beta}}_i = \mathbf{H}_i^\dagger \mathbf{T}$;
- (7) Compute the classification or regression results $\mathbf{O}_i = \mathbf{H}_i \hat{\boldsymbol{\beta}}_i$;
- (8) Randomly generate projection weight $\mathbf{W}_{2i} \in \mathcal{R}^{c \times d}$;
- (9) Compute \mathbf{X}_{i+1} by Eq. (5): $\mathbf{X}_{i+1} = \sigma(\mathbf{X} + \alpha \mathbf{O}_i \mathbf{W}_{2i})$;
- (10) **end**
- (11) Compute the prediction function $f_k(\mathbf{X}) = \mathbf{X}_k \mathbf{W}_{1k} \hat{\boldsymbol{\beta}}_k$.

数据集和比较算法

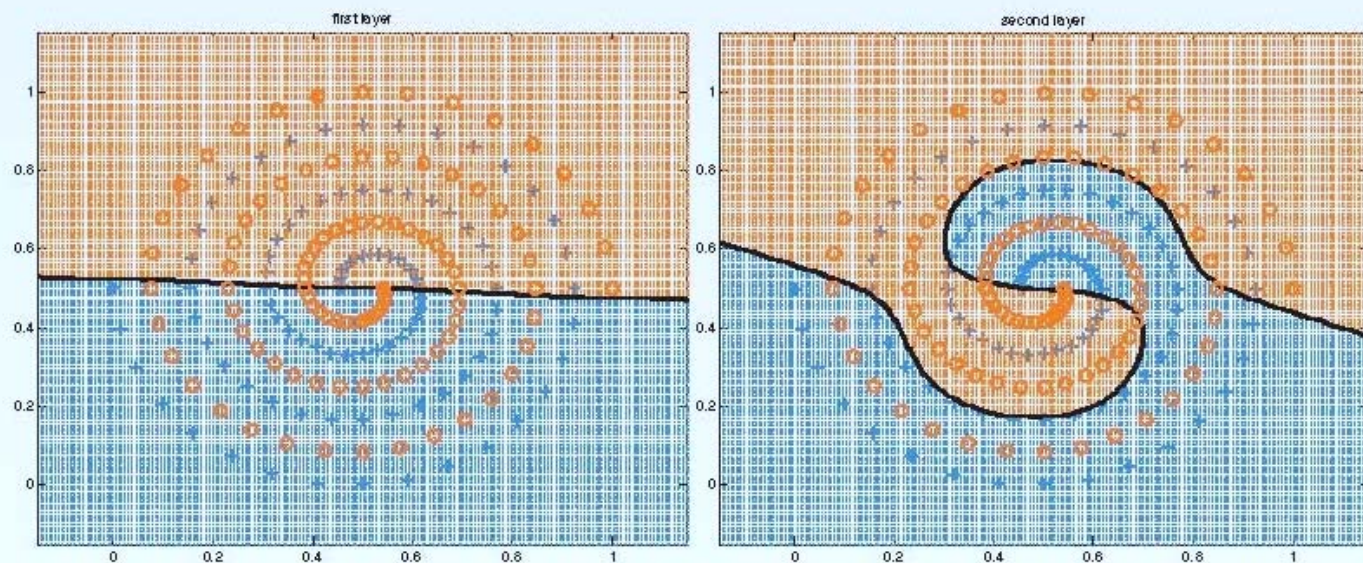
- 数据集：
MNIST、CIFAR-10 和来自 UCI repository 以及 StatLib 的 15 个数据集。
- 比较算法：
 - ① Basic ELM
 - ② Kernel based ELM
 - ③ Stacked Auto-encoder

模拟数据集上的结果



图：XOR 问题的分类超曲面

模拟数据集上的结果



图：Spiral 数据集上的分类超曲面

分类正确率

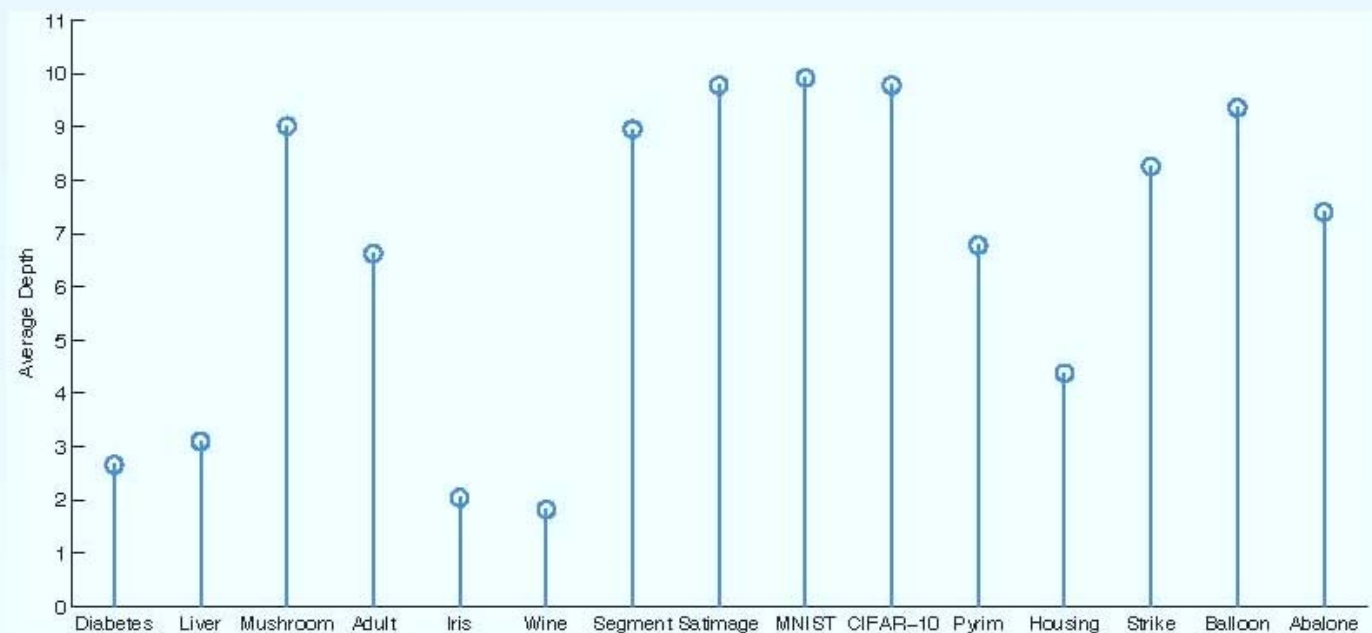
#	Data Sets	ELM	Kernel based ELM		Stacked Autoencoder	DrELM	DrELM ^r
			Gaussian	Sigmoid			
1	Diabetes	0.7461	0.7684	0.7697	0.7773	0.7791	0.7850
2	Liver	0.6609	0.6776	0.6800	0.6957	0.7527	0.7384
3	Mushroom	0.9409	0.9626	0.9751	0.9979	0.9969	0.9842
4	Adult	0.8430	0.8453	0.8460	0.8406	0.8464	0.8495
5	Iris	0.8200	0.9500	0.9480	0.9640	0.9516	0.9607
6	Wine	0.9831	0.9705	0.9834	0.9852	0.9861	0.9785
7	Segment	0.8675	0.9458	0.9444	0.9532	0.9498	0.9544
8	Satimage	0.7762	0.8599	0.8686	0.8743	0.8505	0.8768
9	MNIST	0.8525	0.9211	0.9176	0.9677	0.9463	0.9520
10	CIFAR-10	0.4034	0.3950	0.4287	0.4327	0.4261	0.4331
	Average	0.7894	0.8296	0.8361	0.8489	0.8486	0.8513

回归正确率

表：不同算法在回归数据集的正确率

#	Data Sets	ELM	Kernel based ELM		Stacked Auto-encoder	DrELM	DrELM ^r
			Gaussian	Sigmoid			
1	Pyrim	0.1555	0.1320	0.1281	0.1263	0.1231	0.1290
2	Housing	0.1061	0.0866	0.0851	0.0820	0.0843	0.0791
3	Strike	0.0991	0.0984	0.0987	0.0938	0.0927	0.0940
4	Balloon	0.0143	0.0134	0.0126	0.0105	0.0127	0.0124
5	Abalone	0.0958	0.0876	0.0801	0.0747	0.0749	0.0733
	Average	0.0942	0.0836	0.0809	0.0775	0.0775	0.0776

DrELM 的深度



图：DrELM 取得最高预测准确率时的模型深度

参考文献

- ▶ Wenchao Yu, Fuzhen Zhuang, Qing He and Zhongzhi Shi. Learning Deep Representations via Extreme Learning Machine, **Neurocomputing**, Volume 149, Part A, 3 February 2015, Pages 308-315

谢谢！

