Pattern Anal Applic (2017) 20:383–400
DOI 10.1007/s10044-015-0497-8

THEORETICAL ADVANCES

Extreme entropy machines: robust information theoretic classification

Wojciech Marian Czarnecki[1] · **Jacek Tabor**[1]

Received: 11 February 2015 / Accepted: 7 June 2015 / Published online: 16 July 2015

Abstract Most existing classification methods are aimed at minimization of empirical risk (through some simple point-based error measured with loss function) with added regularization. We propose to approach the classification problem by applying entropy measures as a model objective function. We focus on quadratic Renyi's entropy and connected Cauchy–Schwarz Divergence which leads to the construction of extreme entropy machines (EEM). The main contribution of this paper is proposing a model based on the information theoretic concepts which on the one hand shows new, entropic perspective on known linear classifiers and on the other leads to a construction of very robust method competitive with the state of the art non-information theoretic ones (including Support Vector Machines and Extreme Learning Machines). Evaluation on numerous problems spanning from small, simple ones from UCI repository to the large (hundreds of thousands of samples) extremely unbalanced (up to 100:1 classes' ratios) datasets shows wide applicability of the EEM in real-life problems. Furthermore, it scales better than all considered competitive methods.

Keywords Rapid learning · Extreme learning machines · Classification · Random projections · Entropy

✉ Wojciech Marian Czarnecki
wojciech.czarnecki@uj.edu.pl

Jacek Tabor
jacek.tabor@uj.edu.pl

[1] Faculty of Mathematics and Computer Science, Jagiellonian University, prof. Stanislawa Lojasiewicza 6, Krakow, Poland

1 Introduction

There is no one, universal, perfect optimization criterion that can be used to train machine learning model. Even for linear classifiers, one can find multiple objective functions, error measures to minimize, regularization methods to include [15]. Most existing classification methods are aimed at minimization of empirical risk (through some simple point-based error measured with loss function) with added regularization. We propose to approach this problem in more information theoretic way by investigating applicability of entropy measures as a classification model objective function. We focus on quadratic Renyi's entropy and connected Cauchy–Schwarz Divergence.

One of the information theoretic concepts which has been found very effective in machine learning is the entropy measure. In particular, the rule of maximum entropy modeling led to the construction of MaxEnt model and its structural generalization—Conditional Random Fields which are considered state of the art in many applications. In this paper, we propose to use Renyi's quadratic cross entropy as the measure of two densities' estimations divergence to find the best linear classifier. It is a conceptually different approach than typical entropy models as it works in the input space instead of decisions distribution. As a result, we obtain a model closely related to the Fisher Discriminant (or more generally Linear Discriminant Analysis) which deepens the understanding of this classical approach. Together with a powerful extreme data transformation, we obtain a robust, nonlinear model competitive with the state of the art models not based on information theory such as Support Vector Machines (SVM [4]), Extreme Learning Machines (ELM [11]) or Least Squares Support Vector Machines (LS-SVM [23]). We also show that under some simplifying assumptions

ELM and LS-SVM can be seen through a perspective of information theory as their solutions are (up to some constants) identical to the ones obtained by proposed method.

To draw the general idea of proposed method, we shortly summarize it here. Figure 1 is provided for better intuition. We begin with data in the input space $\mathcal{X}$, transform it randomly into Hilbert space $\mathcal{H}$ where we model them as Gaussians, then perform optimization leading to the projection on $\mathbb{R}$ through $\boldsymbol{\beta}$ and perform density-based classification leading to nonlinear decision boundary in $\mathcal{X}$. It is worth noting that as a result we obtain model that:

- has a trivial implementation (under 20 lines of code in Python),
- learns rapidly,
- is well suited for unbalanced problems,
- constructs nonlinear hypothesis,
- scales very well (better than SVM, LS-SVM and ELM),
- has a few hyperparameters which are easy to optimize.

Paper is structured as follows: first, we recall some preliminary information regarding ELMs and Support Vector Machines, including Least Squares Support Vector Machines. Next, we introduce extreme entropy machine (EEM) together with its kernelized extreme counterpart—extreme entropy kernel machine (EEKM). We show some connections with existing models and some different perspectives for looking at proposed model. In particular, we show how learning capabilities of EEMs (and EEKM) resemble those of ELM (and LS-SVM, respectively). During evaluation on over 20 binary datasets (of various sizes and characteristics), we analyze generalization capabilities and learning speed. We show that it can be a valuable, robust alternative for existing methods. In particular, we show that it achieves analogous of ELM stability in terms of the hidden layer size. We conclude with future development plans and open problems.

2 Preliminaries

Let us begin with recalling some basic information regarding extreme learning machines [12] and Support Vector Machines [4] which are further used as a competing models for proposed solution. We focus here on the optimization problems being solved to underline some basic differences between these methods and EEMs.

2.1 Extreme learning machines

Extreme Learning Machines are relatively young models introduced by Huang et al. [11] which are based on the idea that single layer feed forward neural networks (SLFN) can be trained without iterative process by performing linear regression on the data mapped through random, nonlinear projection (random hidden neurons). More precisely speaking, basic ELM architecture consists of d input neurons connected with each input space dimension which are fully connected with h hidden neurons by the set of weights $\mathbf{w}_j$ (selected randomly from some arbitrary distribution) and set of biases b_j (also randomly selected). Given some generalized nonlinear activation function G, one can express the hidden neurons activation matrix $\mathbf{H}$ for the whole training set $\{(\mathbf{x}_i, t_i)\}_{i=1}^N$ such that $\mathbf{x}_i \in \mathbb{R}^d$ and $t_i \in \{-1, +1\}$ and formulate following optimization problem

2.1.1 Optimization problem: extreme learning machine

$$\begin{aligned} \underset{\boldsymbol{\beta}}{\text{minimize}} \quad & \|\mathbf{H}\boldsymbol{\beta} - \mathbf{t}\|^2 \\ \text{where} \quad & \mathbf{H}_{ij} = \mathrm{G}(\mathbf{x}_i, \mathbf{w}_j, b_j), \quad i = 1, \ldots, N, \quad j = 1, \ldots, h. \end{aligned}$$

If we denote the weights between hidden layer and output neurons by $\boldsymbol{\beta}$ it is easy to show [12] that putting

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{t},$$

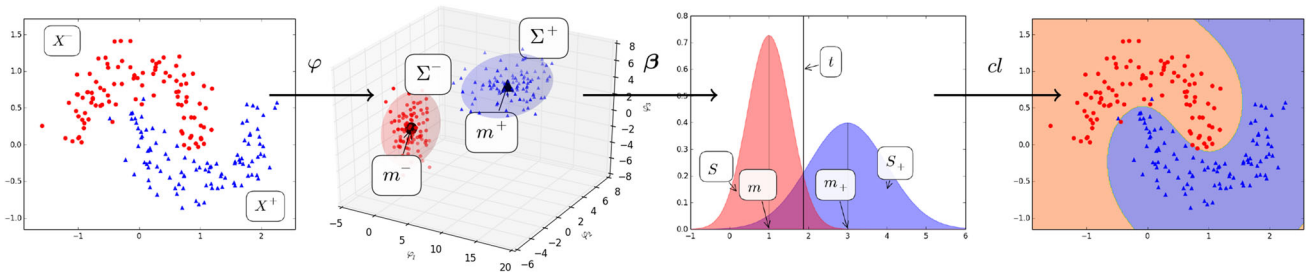

Fig. 1 Visualization of the whole EEM classification process. From the *left* linearly non separable data in $\mathcal{X}$; data mapped to the $\mathcal{H}$ space, where we find covariance estimators; density of projected Gaussians on which the decision is based; decision boundary in the input space $\mathcal{X}$

gives the best solution in terms of mean squared error of the regression:

$$\|\mathbf{H}\boldsymbol{\beta}-\mathbf{t}\|^2=\left\|\mathbf{H}(\mathbf{H}^{\dagger}\mathbf{t})-\mathbf{t}\right\|^2=\min_{\boldsymbol{\alpha}\in\mathbb{R}^h}\|\mathbf{H}\boldsymbol{\alpha}-\mathbf{t}\|^2,$$

where $\mathbf{H}^{\dagger}$ denotes the Moore–Penrose pseudoinverse of $\mathbf{H}$.

Final classification of the new point $\mathbf{x}$ can be now performed analogously by classifying according to

$$cl(\mathbf{x})=\operatorname{sign}([\,\mathrm{G}(\mathbf{x},\mathbf{w}_1,b_1)\quad\ldots\quad\mathrm{G}(\mathbf{x},\mathbf{w}_h,b_h)\,]\boldsymbol{\beta}).$$

As it is based on the ordinary least squares optimization, it is possible to balance it in terms of unbalanced datasets by performing weighted ordinary least squares. In such a scenario, given a vector $\mathbf{s}$ such that $\mathbf{s}_i$ is a square root of the inverse of the $\mathbf{x}_i$'s class size and $\mathbf{s}\cdot\mathbf{X}$ denotes element-wise multiplication between $\mathbf{s}$ and $\mathbf{X}$:

$$\boldsymbol{\beta}=(\mathbf{s}\cdot\mathbf{H})^{\dagger}\mathbf{s}\cdot\mathbf{t}.$$

2.2 Support vector machines and least squares support vector machines

One of the most well-known classifiers of the last decade is Vapnik's support vector machine [4], based on the principle of creating a linear classifier that maximizes the separating margin between elements of two classes.

2.2.1 Optimization problem: support vector machine

$$\begin{aligned}\underset{\boldsymbol{\beta},b,\xi}{\text{minimize}}\quad & \frac{1}{2}\|\boldsymbol{\beta}\|^2+C\sum_{i=1}^{N}\xi_i\\ \text{subject to}\quad & t_i(\langle\boldsymbol{\beta},\mathbf{x}_i\rangle+b)\geq 1-\xi_i\\ & \xi_i\geq 0,\quad i=1,\ldots,N.\end{aligned}$$

here, C denotes the tradeoff between fitting to the data (minimization of the empirical risk) and a size of the separating margin (minimization of model complexity). One needs to fit this hyperparameter to get the best generalization results. SVM can be further kernelized (delinearized) using any kernel K (valid in the Mercer's sense) which leads to the following optimization problem.

2.2.2 Optimization problem: kernel support vector machine

$$\begin{aligned}\underset{\boldsymbol{\beta}}{\text{maximize}}\quad & \sum_{i=1}^{N}\boldsymbol{\beta}_i-\frac{1}{2}\sum_{i,j=1}^{N}\boldsymbol{\beta}_i\boldsymbol{\beta}_j t_i t_j\mathrm{K}(\mathbf{x}_i,\mathbf{x}_j)\\ \text{subject to}\quad & \sum_{i=1}^{N}\boldsymbol{\beta}_i t_i=0\\ & 0\leq\boldsymbol{\beta}_i\leq C,\quad i=1,\ldots,N.\end{aligned}$$

This is a quadratic optimization problem with linear constraints, which can be efficiently solved using quadratic programming techniques. Due to the use of hinge loss function on ξ_i, SVM attains very sparse solutions in terms of nonzero $\boldsymbol{\beta}_i$. As a result, classifier does not have to remember the whole training set, but instead, the set of so-called support vectors ($\mathrm{SV}=\{\mathbf{x}_i:\boldsymbol{\beta}_i>0\}$). A point $\mathbf{x}$ is classified according to

$$cl(\mathbf{x})=\operatorname{sign}\left(\sum_{\mathbf{x}_i\in\mathrm{SV}}t_i\boldsymbol{\beta}_i\mathrm{K}(\mathbf{x}_i,\mathbf{x})+b\right).$$

It appears that if we change the loss function to the quadratic one, we can greatly reduce the complexity of the resulting optimization problem, leading to the so-called Least Squares Support Vector Machines (LS-SVM).

2.2.3 Optimization problem: least squares support vector machine

$$\begin{aligned}\underset{\boldsymbol{\beta},b,\xi}{\text{minimize}}\quad & \frac{1}{2}\|\boldsymbol{\beta}\|^2+C\sum_{i=1}^{N}\xi_i^2\\ \text{subject to}\quad & t_i(\langle\boldsymbol{\beta},\mathbf{x}_i\rangle+b)\geq 1-\xi_i\\ & \xi_i\geq 0,\quad i=1,\ldots,N.\end{aligned}$$

and decision is made according to

$$cl(\mathbf{x})=\operatorname{sign}(\langle\boldsymbol{\beta},\mathbf{x}\rangle+b).$$

As Suykens et al. showed [23] this can be further generalized for arbitrary kernel-induced spaces, where we classify according to:

$$cl(\mathbf{x})=\operatorname{sign}\left(\sum_{i=1}^{N}t_i\boldsymbol{\beta}_i\mathrm{K}(\mathbf{x}_i,\mathbf{x})+b\right),$$

where $\boldsymbol{\beta}_i$ are Lagrange multipliers associated with particular training examples $\mathbf{x}_i$ and b is a threshold, found by solving the linear system

$$\begin{bmatrix}0 & \mathbb{1}^{\mathrm{T}}\\ \mathbb{1} & \mathrm{K}(\mathbf{X},\mathbf{X})+\mathbf{I}/C\end{bmatrix}\begin{bmatrix}b\\ \boldsymbol{\beta}\end{bmatrix}=\begin{bmatrix}0\\ \mathbf{t}\end{bmatrix}$$

where $\mathbb{1}$ is a vector of ones and $\mathbf{I}$ is an identity matrix of appropriate dimensions. Thus, a training procedure becomes

$$\begin{bmatrix}b\\ \boldsymbol{\beta}\end{bmatrix}=\begin{bmatrix}0 & \mathbb{1}^{\mathrm{T}}\\ \mathbb{1} & \mathrm{K}(\mathbf{X},\mathbf{X})+\mathbf{I}/C\end{bmatrix}^{-1}\begin{bmatrix}0\\ \mathbf{t}\end{bmatrix}.$$

Similar to the classical SVM, this formulation is highly unbalanced (its results are skewed towards bigger classes). To overcome this issue, one can introduce a weighted version [24], using diagonal matrix of weights $\mathbf{Q}$, such that $\operatorname{diag}(\mathbf{Q})_i$ is inversely proportional to the size of $\mathbf{x}_i$'s class and

$$\begin{bmatrix} b \\ \boldsymbol{\beta} \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{1}\mathbf{1}^{\mathrm{T}} \\ \mathbf{1}\mathbf{1} & \mathrm{K}(\mathbf{X},\mathbf{X}) + \mathbf{Q}/C \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ \mathbf{t} \end{bmatrix}.$$

Unfortunately, due to the introduction of the square loss, the Support Vector Machines sparseness of the solution is completely lost. Resulting training has a closed-form solution, but requires the computation of the whole Gram matrix and the resulting machine has to remember[1] whole training set to perform new point's classification.

3 Extreme entropy machines

Let us first recall the formulation of the linear classification problem in the highly dimensional feature spaces, i.e., when number of samples N is equal (or less) than dimension of the data space, $\dim(\mathcal{H})$. In particular, we formulate the problem in the limiting case[2] when $\dim(\mathcal{H}) = \infty$:

Problem 1 We are given finite number of (often linearly independent) points $\mathbf{h}^{\pm}$ in an infinite dimensional Hilbert space $\mathcal{H}$. Points $\mathbf{h}^{+} \in \mathbf{H}^{+}$ constitute the positive class, while $\mathbf{h}^{-} \in \mathbf{H}^{-}$ the negative class.

We search for $\boldsymbol{\beta} \in \mathcal{H}$ such that the sets $\boldsymbol{\beta}^{\mathrm{T}}\mathbf{H}^{+}$ and $\boldsymbol{\beta}^{\mathrm{T}}\mathbf{H}^{-}$ are *optimally separated*.

Observe that in itself (without additional regularization) the problem is not well posed as, by applying the linear independence of the data, for arbitrary $m_{+} \neq m_{-}$ in $\mathbb{R}$ we can easily construct $\boldsymbol{\beta} \in \mathcal{H}$ such that

$$\boldsymbol{\beta}^{\mathrm{T}}\mathbf{H}^{+} = \{m_{+}\} \quad \text{and} \quad \boldsymbol{\beta}^{\mathrm{T}}\mathbf{H}^{-} = \{\mathrm{m}_{-}\}. \tag{1}$$

However, this leads to a model case of overfitting, which typically yields suboptimal results on the testing set (different from the original training samples).

To make the problem well posed, we typically need to:

1. add/allow some error in the data,
2. specify some objective function including term penalizing model's complexity.

Popular choices of the objective function include per-point classification loss (like square loss in neural networks or hinge loss in SVM) with a regularization term added, often expressed as the square of the norm of our operator $\boldsymbol{\beta}$ (like in SVM or in weight decay regularization of neural networks). In general, one can divide objective functions derivations into following categories:

- regression based (like in neural networks or ELM),
- probabilistic (like in the case of Naive Bayes),
- geometric (like in SVM),
- information theoretic (entropy models).

We focus on the last group of approaches, and investigate the applicability of the *Cauchy–Schwarz Divergence* [13], which for two densities f and g is given by

$$\begin{aligned} D_{\mathrm{CS}}(f,g) &= \ln\left(\int f^2\right) + \ln\left(\int g^2\right) - 2\ln\left(\int fg\right) \\ &= -2\ln\left(\int \frac{f}{\|f\|_2}\frac{g}{\|g\|_2}\right). \end{aligned}$$

Cauchy–Schwarz Divergence is connected to Renyi's quadratic cross entropy ($H_2^{\times}$ [21]) and Renyi's quadratic entropy (H_2), defined for densities f, g as

$$\begin{aligned} H_2^{\times}(f,g) &= -\ln\left(\int fg\right) \\ H_2(f) &= H_2^{\times}(f,f) = -\ln\left(\int f^2\right), \end{aligned}$$

consequently

$$D_{\mathrm{CS}}(f,g) = 2H_2^{\times}(f,g) - H_2(f) - H_2(g)$$

and as showed in [7], it is well suited as a discrimination measure which allows the construction of multi-threshold linear classifiers. In general, increase of the value of Cauchy–Schwarz Divergence results in better sets' (densities') discrimination. Unfortunately, there are a few problems with such an approach:

- true datasets are discrete, so we do not know actual densities f and g,
- statistical density estimators require rather large sample sizes and are very computationally expensive.

There are basically two approaches which help us recover underlying densities from the samples. First one is performing some kind of density estimation, like the well-known Kernel Density Estimation (KDE) technique, which is based on the observation that any arbitrary continuous distribution can be approximated arbitrarily closely by the convex combination of Gaussians [26]. The other approach is to assume some density model (distribution's family) and fit its parameters to maximize the data generation probability. In statistics, it is known as maximum likelihood estimation (MLE) approach. MLE has an advantage that in general it produces much simpler densities descriptions than KDE as the second one's description scales linearly in terms of sample size.

A common choice of density models are Gaussian distributions due to their nice theoretical and practical (computational) capabilities. As mentioned earlier, the convex combination of Gaussians can approximate the given

[1] There are some pruning techniques for LS-SVM but we are not investigating them here.

[2] Which is often obtained by the kernel approach.

continuous distribution f with arbitrary precision. To fit a Gaussian mixture model (GMM) to given dataset, one needs an algorithm such as Expectation Maximization [8] or conceptually similar Cross-entropy clustering [25]. However, for simplicity and strong regularization, we propose to model f as one big Gaussian $\mathcal{N}(\mathbf{m}, \Sigma)$. One of the biggest advantages of such an approach is closed-form MLE parameter estimation, as we simply put $\mathbf{m}$ equal to the empirical mean of the data, and Σ as some data covariance estimator. Second, this way we introduce an error to the data which has an important regularizing role and leads to better posed optimization problem.

Let us recall that the Shannon's differential entropy (expressed in nits) of the continuous distribution f is

$$H(f) = -\int f \ln f.$$

We will show that choice of normal distributions is not arbitrary but supported by the assumptions of the entropy maximization. Following result is known [5], but we include the whole reasoning for completeness.

Lemma 1 *Normal distribution $\mathcal{N}(\mathbf{m}, \Sigma)$ has a maximum Shannon's differential entropy among all real-valued distributions with mean $\mathbf{m} \in \mathbb{R}^h$ and covariance $\Sigma \in \mathbb{R}^{h\times h}$.*

Proof Let f and g be arbitrary distributions with covariance Σ and means $\mathbf{m}$. For simplicity, we assume that $\mathbf{m} = 0$ but the analogous proof holds for arbitrary mean, then

$$\int f \mathbf{h}_i \mathbf{h}_j d\mathbf{h}_i d\mathbf{h}_j = \int g \mathbf{h}_i \mathbf{h}_j d\mathbf{h}_i d\mathbf{h}_j = \Sigma_{ij},$$

so for quadratic form $\mathbf{A}$

$$\int \mathbf{A} f = \int \mathbf{A} g.$$

Notice that

$$\ln \mathcal{N}(0,\Sigma)[\mathbf{h}] = \ln\left(\frac{1}{\sqrt{(2\pi)^h \det(\Sigma)}} \exp(-\tfrac{1}{2}\mathbf{h}^{\mathrm{T}}\Sigma^{-1}\mathbf{h})\right)$$
$$= -\frac{1}{2}\ln((2\pi)^h \det(\Sigma)) - \tfrac{1}{2}\mathbf{h}^{\mathrm{T}}\Sigma^{-1}\mathbf{h}$$

is a quadratic form plus constant. Thus,

$$\int f \ln \mathcal{N}(0,\Sigma) = \int \mathcal{N}(0,\Sigma) \ln \mathcal{N}(0,\Sigma),$$

which together with non-negativity of Kullback–Leibler Divergence gives

$$\begin{aligned} 0 &\le D_{\mathrm{KL}}(f \,||\, \mathcal{N}(0,\Sigma)) \\ &= \int f \ln\left(\tfrac{f}{\mathcal{N}(\prime,\Sigma)}\right) \\ &= \int f \ln f - \int f \ln \mathcal{N}(0,\Sigma) \\ &= -H(f) - \int f \ln \mathcal{N}(0,\Sigma) \\ &= -H(f) - \int \mathcal{N}(0,\Sigma) \ln \mathcal{N}(0,\Sigma) \\ &= -H(f) + H(\mathcal{N}(0,\Sigma)), \end{aligned}$$

Consequently, $H(\mathcal{N}(0,\Sigma)) \ge H(f)$. □

There appears nontrivial question how to find/estimate the desired Gaussian as the covariance can be singular. In this case, to regularize the covariance, we apply the well-known Ledoit–Wolf approach [16]

$$\Sigma^{\pm} = \mathrm{cov}_{\dagger}(\mathbf{H}^{\pm}) = (1-\varepsilon^{\pm})\mathrm{cov}(\mathbf{H}^{\pm}) + \varepsilon^{\pm}\,\mathrm{tr}(\mathrm{cov}(\mathbf{H}^{\pm}))h^{-1}\mathbf{I},$$

where $\mathrm{cov}(\cdot)$ is an empirical covariance and $\varepsilon^{\pm}$ is a shrinkage coefficient given in closed form by Ledoit and Wolf [16]. We would like to underline that this estimator is parameterless and is optimal in the sense that it minimizes the expected quadratic loss between true covariance matrix and the estimator under general asymptotics (meaning that both dataset size and its dimension grow to infinity, but their ratio converges to a constant).

Thus, our optimization problem can be stated as follows:

Problem 2 (*Optimization problem*) Suppose that we are given two datasets $\mathbf{H}^{\pm}$ in a Hilbert space $\mathcal{H}$ which come from the Gaussian distributions $\mathcal{N}(\mathbf{m}^{\pm}, \Sigma^{\pm})$. Find $\boldsymbol{\beta} \in \mathcal{H}$ such that the datasets

$$\boldsymbol{\beta}^{\mathrm{T}}\mathbf{H}^{+} \quad \text{and} \quad \boldsymbol{\beta}^{\mathrm{T}}\mathbf{H}^{-}$$

are optimally discriminated in terms of Cauchy–Schwarz Divergence.

Because $\mathbf{H}^{\pm}$ has density $\mathcal{N}(\mathbf{m}^{\pm}, \Sigma^{\pm})$, we know that $\boldsymbol{\beta}^{\mathrm{T}}\mathbf{X}^{\pm}$ has the density $\mathcal{N}(\boldsymbol{\beta}^{\mathrm{T}}\mathbf{m}^{\pm}, \boldsymbol{\beta}^{\mathrm{T}}\Sigma^{\pm}\boldsymbol{\beta})$. We put

Table 1 Spearman's rank correlation coefficient between optimized term and whole D_{CS} for all datasets used in evaluation

Dataset	1	10	100	200	500
AUSTRALIAN	0.928	−0.022	0.295	0.161	0.235
BREAST-CANCER	0.628	0.809	0.812	0.858	0.788
DIABETES	−0.983	−0.976	−0.941	−0.982	−0.952
GERMAN.NUMER	0.916	0.979	0.877	0.873	0.839
HEART	0.964	0.829	0.931	0.91	0.893
IONOSPHERE	0.999	0.988	0.98	0.978	0.984
LIVER DISORDERS	0.232	0.308	0.363	0.33	0.312
SONAR	−0.31	−0.542	−0.41	−0.407	−0.381
SPLICE	−0.284	−0.036	−0.165	−0.118	−0.101
ABALONE7	1.0	0.999	0.999	0.999	0.998
ARYTHMIA	1.0	1.0	0.999	1.0	1.0
BALANCE	1.0	0.998	0.998	0.999	0.998
CAR EVALUATION	1.0	0.998	0.998	0.997	0.997
ECOLI	0.964	0.994	0.995	0.998	0.995
LIBRAS MOVE	1.0	0.999	0.999	1.0	1.0
OIL SPILL	1.0	1.0	1.0	1.0	1.0
SICK EUTHYROID	1.0	0.999	1.0	1.0	1.0
SOLAR FLARE	1.0	1.0	1.0	1.0	1.0
SPECTROMETER	1.0	1.0	0.999	0.999	0.999
FOREST COVER	0.988	0.997	0.997	0.992	0.988
ISOLET	0.784	1.0	0.997	0.997	0.999
MAMMOGRAPHY	1.0	1.0	1.0	1.0	1.0
PROTEIN HOMOLOGY	1.0	1.0	1.0	1.0	1.0
WEBPAGES	1.0	1.0	1.0	0.999	0.999

Each column represents different dimension of the Hilbert space

$$m_{\pm} = \boldsymbol{\beta}^{\mathrm{T}}\mathbf{m}^{\pm},\ S_{\pm} = \boldsymbol{\beta}^{\mathrm{T}}\Sigma^{\pm}\boldsymbol{\beta}. \tag{2}$$

Since, as one can easily compute [6],

$$\begin{aligned}\int \frac{\mathcal{N}(m_+,S_+)}{\|\mathcal{N}(m_+,S_+)\|_2}\cdot\frac{\mathcal{N}(m_-,S_-)}{\|\mathcal{N}(m_-,S_-)\|_2} &= \frac{\mathcal{N}(m_+-m_-,S_++S_-)[0]}{(\mathcal{N}(0,2S_+)[0]\mathcal{N}(0,2S_-)[0])^{1/2}}\\ &= \frac{(2\pi S_+S_-)^{1/4}}{(S_++S_-)^{1/2}}\exp\left(-\frac{(m_+-m_-)^2}{2(S_++S_-)}\right),\end{aligned}$$

we obtain that

$$\begin{aligned}&D_{\mathrm{CS}}(\mathcal{N}(m_+,S_+),\mathcal{N}(m_-,S_-))\\ &\quad= -\ln\left(\int \frac{\mathcal{N}(m_+,S_+)}{\|\mathcal{N}(m_+,S_+)\|_2}\cdot\frac{\mathcal{N}(m_-,S_-)}{\|\mathcal{N}(m_-,S_-)\|_2}\right)\\ &\quad= -\frac{1}{2}\ln\frac{\pi}{2} - \ln\frac{\frac{1}{2}(S_++S_-)}{\sqrt{S_+S_-}} + \frac{(m_+-m_-)^2}{S_++S_-}.\end{aligned} \tag{3}$$

Observe that in the above equation the first term is constant, the second is the logarithm of the quotient of arithmetical and geometrical means and therefore in the typical cases is bounded and close to zero. However, in some singular cases, when S_+ and S_- are of different order of magnitude the above term can blow up, and thus more precisely here we are maximizing a lower bound of D_{CS}. As we will see in further sections, this simplification leads to the closed-form solution, which in particular means that learning process is extremely simple. In general, one could maximize the whole function using gradient methods, but this would lead to the need of introducing optimization procedure hyper-parameters (such as learning rate, momentum, stopping condition, etc.) which we are trying to avoid.

Consequently, crucial information is given by the last term. To confirm this claim, we perform experiments on over 20 datasets used in further evaluation (more details are located in the Evaluation section). We compute the Spearman's rank correlation coefficient between the $D_{\mathrm{CS}}(\mathcal{N}(m_+,S_+),\mathcal{N}(m_-,S_-))$ and $\frac{(m_+-m_-)^2}{S_++S_-}$ for hundred random projections to $\mathcal{H}$ and hundred random linear operators $\boldsymbol{\beta}$. As one can see in Table 1, in small datasets (first part of the table) the correlation is generally high, with some exceptions (like SONAR, SPLICE, LIVER DISORDERS and DIABETES). However, for bigger datasets (consisting of thousands examples), this correlation is nearly perfect (up to the randomization process it is nearly 1.0 for all cases) which is a very strong empirical confirmation of our claim that maximization of the $\frac{(m_+-m_-)^2}{S_++S_-}$ is generally equivalent to the maximization of $D_{\mathrm{CS}}(\mathcal{N}(m_+,S_+),\mathcal{N}(m_-,S_-))$ as correlation coefficient captures if D_{CS} and its lower bound have similar monotonicity. Number of dimensions of the Hilbert space does not affect the result in a significant manner for large datasets, while in the case of small ones it seems that results vary significantly when it is changed. This might be the consequence of less reliable covariance estimations for small datasets, especially with higher number of dimensions.

This means that, after the above reductions, and application of (2) our final problem can be stated as follows:

3.1 Optimization problem: extreme entropy machine

$$\begin{aligned}\underset{\boldsymbol{\beta}}{\text{minimize}}\quad & \boldsymbol{\beta}^{\mathrm{T}}\Sigma^{+}\boldsymbol{\beta} + \boldsymbol{\beta}^{\mathrm{T}}\Sigma^{-}\boldsymbol{\beta}\\ \text{subject to}\quad & \boldsymbol{\beta}^{\mathrm{T}}(\mathbf{m}^{+}-\mathbf{m}^{-}) = 2\\ \text{where}\quad & \Sigma^{\pm} = \mathrm{cov}_{\dagger}(\mathbf{H}^{\pm})\\ & \mathbf{m}^{\pm} = \frac{1}{|\mathbf{H}^{\pm}|}\sum_{\mathbf{h}^{\pm}\in\mathbf{H}^{\pm}}\mathbf{h}^{\pm}\\ & \mathbf{H}^{\pm} = \varphi(\mathbf{X}^{\pm})\end{aligned}$$

Before we continue to the closed-form solution, we outline two methods of actually transforming our data $\mathbf{X}^{\pm} \subset \mathcal{X}$ to the highly dimensional $\mathbf{H}^{\pm} \subset \mathcal{H}$, given by the $\varphi : \mathcal{X} \to \mathcal{H}$.

We investigate two approaches which lead to the Extreme Entropy Machine and Extreme Entropy Kernel Machine, respectively.

- For Extreme Entropy Machine (EEM), we use the random projection technique, exactly the same as the one used in the ELM. In other words, given some generalized activation function $\mathrm{G}(\mathbf{x},\mathbf{w},b):\mathcal{X}\times\mathcal{X}\times\mathbb{R}\rightarrow\mathbb{R}$ and a constant h denoting number of hidden neurons:

$$\varphi:\mathcal{X}\ni\mathbf{x}\rightarrow[\mathrm{G}(\mathbf{x},\mathbf{w}_1,b_1),\ldots,\mathrm{G}(\mathbf{x},\mathbf{w}_h,b_h)]^{\mathrm{T}}\in\mathbb{R}^h$$

  where w_i are random vectors and b_i are random biases.
- For Extreme Entropy Kernel Machine (EEKM), we use the randomized kernel approximation technique [9], which spans our Hilbert space on randomly selected subset of training vectors. In other words, given valid kernel $\mathrm{K}(\cdot,\cdot):\mathcal{X}\times\mathcal{X}\rightarrow\mathbb{R}_+$ and size of the kernel space base h:

$$\varphi_{\mathrm{K}}:\mathcal{X}\ni\mathbf{x}\rightarrow(\mathrm{K}(\mathbf{x},\mathbf{X}^{[h]})\mathrm{K}(\mathbf{X}^{[h]},\mathbf{X}^{[h]})^{-1/2})^{\mathrm{T}}\in\mathbb{R}^h$$

  where $\mathbf{X}^{[h]}$ is a h element random subset of $\mathbf{X}$. It is easy to verify that such low rank approximation truly behaves as a kernel, in the sense that for $\varphi_{\mathrm{K}}(\mathbf{x}_i),\varphi_{\mathrm{K}}(\mathbf{x}_j)\in\mathbb{R}^h$

$$\begin{aligned}
&\varphi_{\mathrm{K}}(\mathbf{x}_i)^{\mathrm{T}}\varphi_{\mathrm{K}}(\mathbf{x}_j)\\
&\quad=((\mathrm{K}(\mathbf{x}_i,\mathbf{X}^{[h]})\mathrm{K}(\mathbf{X}^{[h]},\mathbf{X}^{[h]})^{-1/2})^{\mathrm{T}})^{\mathrm{T}}\\
&\qquad\times(\mathrm{K}(\mathbf{x}_j,\mathbf{X}^{[h]})\mathrm{K}(\mathbf{X}^{[h]},\mathbf{X}^{[h]})^{-1/2})^{\mathrm{T}}\\
&\quad=\mathrm{K}(\mathbf{x}_i,\mathbf{X}^{[h]})\mathrm{K}(\mathbf{X}^{[h]},\mathbf{X}^{[h]})^{-1/2}\\
&\qquad\times(\mathrm{K}(\mathbf{x}_j,\mathbf{X}^{[h]})\mathrm{K}(\mathbf{X}^{[h]},\mathbf{X}^{[h]})^{-1/2})^{\mathrm{T}}\\
&\quad=\mathrm{K}(\mathbf{x}_i,\mathbf{X}^{[h]})\mathrm{K}(\mathbf{X}^{[h]},\mathbf{X}^{[h]})^{-1/2}\\
&\mathrm{K}(\mathbf{X}^{[h]},\mathbf{X}^{[h]})^{-1/2}\mathrm{K}^{\mathrm{T}}(\mathbf{x}_j,\mathbf{X}^{[h]})\\
&\quad=\mathrm{K}(\mathbf{x}_i,\mathbf{X}^{[h]})\mathrm{K}(\mathbf{X}^{[h]},\mathbf{X}^{[h]})^{-1}\mathrm{K}(\mathbf{X}^{[h]},\mathbf{x}_j).
\end{aligned}$$

  Given true kernel projection ϕ_{K} such that $\mathrm{K}(\mathbf{x}_i,\mathbf{x}_j)=\phi_{\mathrm{K}}(\mathbf{x}_i)^{\mathrm{T}}\phi_{\mathrm{K}}(\mathbf{x}_j)$, we have

$$\begin{aligned}
&\mathrm{K}(\mathbf{x}_i,\mathbf{X}^{[h]})\mathrm{K}(\mathbf{X}^{[h]},\mathbf{X}^{[h]})^{-1}\mathrm{K}(\mathbf{X}^{[h]},\mathbf{x}_j)\\
&\quad=\phi_{\mathrm{K}}(\mathbf{x}_i)^{\mathrm{T}}\phi_{\mathrm{K}}(\mathbf{X}^{[h]})\\
&\qquad\times(\phi_{\mathrm{K}}(\mathbf{X}^{[h]})^{\mathrm{T}}\phi_{\mathrm{K}}(\mathbf{X}^{[h]}))^{-1}\\
&\qquad\times\phi_{\mathrm{K}}(\mathbf{X}^{[h]})^{\mathrm{T}}\phi_{\mathrm{K}}(\mathbf{x}_j)\\
&\quad=\phi_{\mathrm{K}}(\mathbf{x}_i)^{\mathrm{T}}\phi_{\mathrm{K}}(\mathbf{X}^{[h]})\phi_{\mathrm{K}}(\mathbf{X}^{[h]})^{-1}\\
&\qquad\times(\phi_{\mathrm{K}}(\mathbf{X}^{[h]})^{\mathrm{T}})^{-1}\phi_{\mathrm{K}}(\mathbf{X}^{[h]})^{\mathrm{T}}\phi_{\mathrm{K}}(\mathbf{x}_j)\\
&\quad=\phi_{\mathrm{K}}(\mathbf{x}_i)^{\mathrm{T}}\phi_{\mathrm{K}}(\mathbf{x}_j)\\
&\quad=\mathrm{K}(\mathbf{x}_i,\mathbf{x}_j).
\end{aligned}$$

  Thus, for the whole samples' set, we have

$$\varphi_{\mathrm{K}}(\mathbf{X})^{\mathrm{T}}\varphi_{\mathrm{K}}(\mathbf{X})=\mathrm{K}(\mathbf{X},\mathbf{X}),$$

  which is a complete Gram matrix.

So the only difference between Extreme Entropy Machine and Extreme Entropy Kernel Machine is that in later we use $\mathbf{H}^{\pm}=\varphi_{\mathrm{K}}(\mathbf{X}^{\pm})$ where K is a selected kernel instead of $\mathbf{H}^{\pm}=\varphi(\mathbf{X}^{\pm})$. Figure 2 visualizes these two approaches as neural networks, in particular EEM is a simple SLFN, while EEKM leads to the network with two hidden layers.

Remark 1 Extreme Entropy Machine optimization problem is closely related to the SVM optimization, but instead of maximizing the margin between closest points we are maximizing the mean margin.

Proof Let us recall that in SVM we try to maximize the margin $\frac{2}{\|\boldsymbol{\beta}\|}$ under constraints that negative samples are projected at values at most -1 ($\boldsymbol{\beta}^{\mathrm{T}}\mathbf{h}^{-}+b\leq-1$) and positive samples on at least 1 ($\boldsymbol{\beta}^{\mathrm{T}}\mathbf{h}^{+}+b\geq1$). In other words, we are minimizing the $\boldsymbol{\beta}$ operator norm $\|\boldsymbol{\beta}\|$ which is equivalent to minimizing the square of this norm $\|\boldsymbol{\beta}\|^2$, under constraint

$$\min_{\mathbf{h}^{+}\in\mathbf{H}^{+}}\{\boldsymbol{\beta}^{\mathrm{T}}\mathbf{h}^{+}\}-\max_{\mathbf{h}^{-}\in\mathbf{H}^{-}}\{\boldsymbol{\beta}^{\mathrm{T}}\mathbf{h}^{-}\}=1-(-1)=2.$$

On the other hand, EEM tries to minimize

$$\begin{aligned}
\boldsymbol{\beta}^{\mathrm{T}}\Sigma^{+}\boldsymbol{\beta}+\boldsymbol{\beta}^{\mathrm{T}}\Sigma^{-}\boldsymbol{\beta}&=\boldsymbol{\beta}^{\mathrm{T}}(\Sigma^{+}+\Sigma^{-})\boldsymbol{\beta}\\
&=\|\boldsymbol{\beta}\|^2_{(\Sigma^{+}+\Sigma^{-})^{-1}}
\end{aligned}$$

under the constraint

$$\frac{1}{|\mathbf{H}^{+}|}\sum_{\mathbf{h}^{+}\in\mathbf{H}^{+}}\boldsymbol{\beta}^{\mathrm{T}}\mathbf{h}^{+}-\frac{1}{|\mathbf{H}^{-}|}\sum_{\mathbf{h}^{-}\in\mathbf{H}^{-}}\boldsymbol{\beta}^{\mathrm{T}}\mathbf{h}^{-}=2.$$

So what is happening here is that we are trying to maximize the mean margin between classes in the Mahalanobis norm [17] generated by the inverse of the sum of classes' covariances. □

Similar observation regarding connection between large margin classification and entropy optimization has been done in case of the Multithreshold Linear Entropy Classifier [7]. One should also notice important relations to other methods studying so-called margin distributions [10], such as Large margin Distribution Machine [29]. Contrary to Zhang et al. approach, we are minimizing the summarized variances instead of minimizing the difference between data variance and cross class variance. As a result, proposed model is much easier to optimize (as shown below).

We are going to show by applying the standard method of Lagrange multipliers that the above problem has a closed-form solution (similar to the Fisher Discriminant). We put

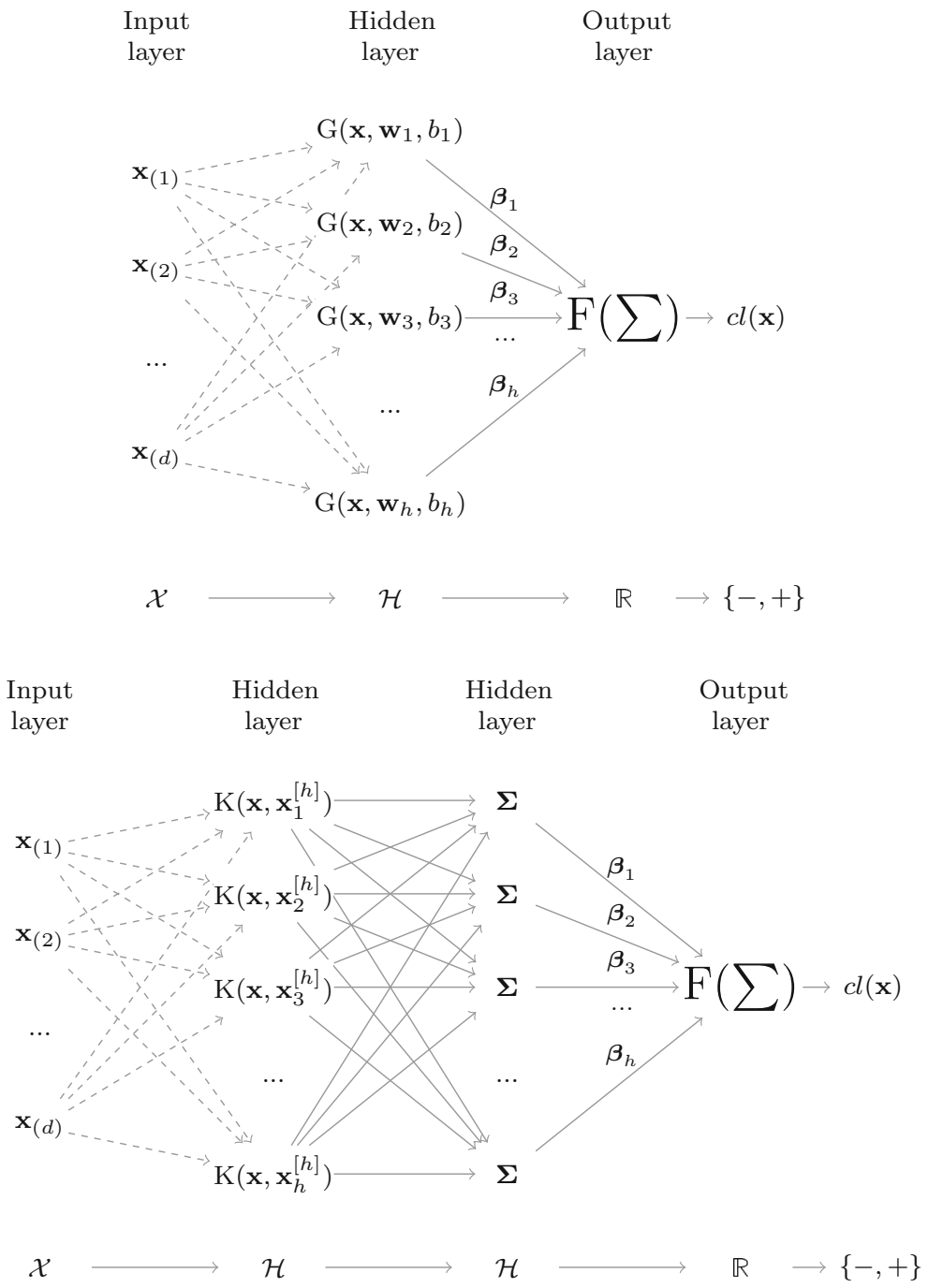

Fig. 2 Extreme entropy machine (*top*) and extreme entropy kernel machine (*bottom*) as neural networks. In both cases, all weights are either randomly selected (*dashed*) or are the result of closed-form optimization (*solid*)

$$L(\boldsymbol{\beta}, \lambda) = 2\boldsymbol{\beta}^{\mathrm{T}}(\Sigma^+ + \Sigma^-)\boldsymbol{\beta} - \lambda(\boldsymbol{\beta}^{\mathrm{T}}(\mathbf{m}^+ - \mathbf{m}^-) - 2).$$

Then,

$$\nabla_{\boldsymbol{\beta}} L = 2(\Sigma^+ + \Sigma^-)\boldsymbol{\beta} - \lambda(\mathbf{m}^+ - \mathbf{m}^-) \quad \text{and}$$
$$\frac{\partial}{\partial \lambda} L = \boldsymbol{\beta}^{\mathrm{T}}(\mathbf{m}^+ - \mathbf{m}^-) - 2,$$

which means that we need to solve, with respect to $\boldsymbol{\beta}$, the system

$$\begin{cases} 2(\Sigma^+ + \Sigma^-)\boldsymbol{\beta} - \lambda(\mathbf{m}^+ - \mathbf{m}^-) = 0, \\ \boldsymbol{\beta}^{\mathrm{T}}(\mathbf{m}^+ - \mathbf{m}^-) = 2. \end{cases}$$

Therefore, $\boldsymbol{\beta} = \frac{\lambda}{2}(\Sigma^+ + \Sigma^-)^{-1}(\mathbf{m}^+ - \mathbf{m}^-)$, which yields

$$\frac{\lambda}{2}(\mathbf{m}^+ - \mathbf{m}^-)^{\mathrm{T}}(\Sigma^+ + \Sigma^-)^{-1}(\mathbf{m}^+ - \mathbf{m}^-) = 2,$$

and consequently,[3] if $\mathbf{m}^+ - \mathbf{m}^- \neq 0$, then $\lambda = 4/\|\mathbf{m}^+ - \mathbf{m}^-\|^2_{\Sigma^+ + \Sigma^-}$ and

$$\begin{aligned}\boldsymbol{\beta} &= \frac{2}{\|\mathbf{m}^+ - \mathbf{m}^-\|^2_{\Sigma^+ + \Sigma^-}}(\Sigma^+ + \Sigma^-)^{-1}(\mathbf{m}^+ - \mathbf{m}^-) \\ &= \frac{2(\Sigma^+ + \Sigma^-)^{-1}(\mathbf{m}^+ - \mathbf{m}^-)}{\|\mathbf{m}^+ - \mathbf{m}^-\|^2_{\Sigma^+ + \Sigma^-}}.\end{aligned} \tag{4}$$

The final decision of the class of the point $\mathbf{h}$ is therefore given by the comparison of the values

$$\mathcal{N}(\boldsymbol{\beta}^{\mathrm{T}}\mathbf{m}^+, \boldsymbol{\beta}^{\mathrm{T}}\Sigma^+\boldsymbol{\beta})[\boldsymbol{\beta}^{\mathrm{T}}\mathbf{h}] \quad \text{and} \quad \mathcal{N}(\boldsymbol{\beta}^{\mathrm{T}}\mathbf{m}^-, \boldsymbol{\beta}^{\mathrm{T}}\Sigma^-\boldsymbol{\beta})[\boldsymbol{\beta}^{\mathrm{T}}\mathbf{h}].$$

We distinguish two cases based on number of resulting classifier's thresholds (points r such that $\mathcal{N}(\boldsymbol{\beta}^{\mathrm{T}}\mathbf{m}^+, \boldsymbol{\beta}^{\mathrm{T}}\Sigma^+\boldsymbol{\beta})[r] = \mathcal{N}(\boldsymbol{\beta}^{\mathrm{T}}\mathbf{m}^-, \boldsymbol{\beta}^{\mathrm{T}}\Sigma^-\boldsymbol{\beta})[r]$):

1. $S_- = S_+$, then there is one threshold
$$r_0 = m_- + 1,$$
which results in a traditional (one-threshold) linear classifier,
2. $S_- \neq S_+$, then there are two thresholds
$$r_\pm = m_- + \frac{2S_- \pm \sqrt{S_-S_+(\ln(S_-/S_+)(S_- - S_+) + 4)}}{S_- - S_+},$$
which makes the resulting classifier a member of two-thresholds linear classifiers family [1].

Obviously, in the degenerated case, when $m = 0 \iff \mathbf{m}^- = \mathbf{m}^+$ there is no solution, as the constraint $\boldsymbol{\beta}^{\mathrm{T}}(\mathbf{m}^- - \mathbf{m}^+) = 2$ is not fulfilled for any $\boldsymbol{\beta}$. In such a case, EEM returns a trivial classifier constantly equal to any class (we put $\boldsymbol{\beta} = 0$).

From the neural network perspective, we simply construct a custom activation function $\mathrm{F}(\cdot)$ in the output neuron depending on one of the two described cases:

1. $\mathrm{F}(x) = \begin{cases} +1, & \text{if } x \geq r_0 \\ -1, & \text{if } x < r_0 \end{cases} = \mathrm{sign}(x - r_0),$
2. $\mathrm{F}(x) = \begin{cases} +1, & \text{if } x \in [r_-, r_+] \\ -1, & \text{if } x \notin [r_-, r_+] \end{cases} = -\mathrm{sign}(x - r_-)\mathrm{sign}(x - r_+),$
if $r_- < r_+$
and
$$\mathrm{F}(x) = \begin{cases} -1, & \text{if } x \in [r_+, r_-] \\ +1, & \text{if } x \notin [r_+, r_-] \end{cases} = \mathrm{sign}(x - r_-)\mathrm{sign}(x - r_+),$$
otherwise.

[3] Where $\|\mathbf{m}^+ - \mathbf{m}^-\|^2_{\Sigma^+ + \Sigma^-} = (\mathbf{m}^+ - \mathbf{m}^-)^{\mathrm{T}}(\Sigma^+ + \Sigma^-)^{-1}(\mathbf{m}^+ - \mathbf{m}^-)$ denotes the squared Mahalanobis norm of $\mathbf{m}^+ - \mathbf{m}^-$.

4 Theory: density estimation in the kernel case

To illustrate our reasoning, we consider a typical basic problem concerning the density estimation.

Problem 3 Assume that we are given a finite dataset $\mathbf{H}$ in a Hilbert space $\mathcal{H}$ generated by the unknown density f, and the goal consists in estimating f.

Since the problem in itself is infinite dimensional, typically the data would be linearly independent [20]. Moreover, one usually cannot obtain reliable density estimation—the most we can hope is that after transformation by a linear functional into $\mathbb{R}$, the resulting density will be well estimated.

To simplify the problem assume therefore that we want to find the desired density in the class of normal densities—or equivalently that we are interested only in the estimation of the mean and covariance of f.

The generalization of the above problem is given by the following problem:

Problem 4 Assume that we are given a finite dataset $\mathbf{h}^\pm$ in a Hilbert space $\mathcal{H}$ generated by the unknown densities $f^\pm$, and the goal consists in estimating the unknown densities.

In general, $\dim(\mathcal{H}) \gg \mathrm{N}$which means that we have very sparse data in terms of Hilbert space. As a result, classical kernel density estimation (KDE) is not reliable source of information [18]. In the absence of different tools, we can however use KDE with very big kernel width to cover at least some general shape of the whole density.

Remark 2 Assume that we are given a finite dataset $\mathbf{h}^\pm$ with means $\mathbf{m}^\pm$ and covariances $\Sigma^\pm$ in a Hilbert space $\mathcal{H}$. If we conduct kernel density estimation using a Gaussian kernel then, in a limiting case, each class becomes a normal distribution. Strictly speaking

$$\lim_{\sigma \to \infty} \|[\![\mathbf{H}^\pm]\!]_\sigma - \mathcal{N}(\mathbf{m}^\pm, \sigma^2\Sigma^\pm)\|_2 = 0,$$

where

$$[\![A]\!]_\sigma = \frac{1}{|A|}\sum_{a \in A}\mathcal{N}(a, \sigma^2 \cdot \mathrm{cov}(A)).$$

Proof of this remark is given by Czarnecki and Tabor [7] and means that if we perform a Gaussian kernel density estimation of our data with big kernel width (which is reasonable for small amount of data in highly dimensional space) then for big enough $\hat{\sigma}$ EEM is nearly optimal linear classifier in terms of estimated densities

$$\hat{f}^\pm = \mathcal{N}(\mathbf{m}^\pm, \hat{\sigma}^2\Sigma^\pm) \approx [\![\mathbf{H}^\pm]\!]_{\hat{\sigma}}.$$

Let us now investigate the probabilistic interpretation of EEM. Under the assumption that $\mathbf{H}^{\pm} \sim \mathcal{N}(\mathbf{m}^{\pm}, \Sigma^{\pm})$, we have the conditional probabilities

$$p(\mathbf{h}|\pm) = \mathcal{N}(\mathbf{m}^{\pm}, \Sigma^{\pm})[\mathbf{h}],$$

so from Bayes rule we conclude that

$$p(\pm|\mathbf{h}) = \frac{p(\mathbf{h}|\pm)p(\pm)}{p(\mathbf{h})} \propto \mathcal{N}(\mathbf{m}^{\pm}, \Sigma^{\pm})[\mathbf{h}]p(\pm),$$

where $p(\pm)$ is a prior classes' distribution. In our case, due to the balanced nature (meaning that despite classes imbalance we maximize the balanced quality measure such as Balanced Accuracy), we have $p(\pm) = 1/2$.

But

$$p(\mathbf{h}) = \sum_{t\in\{+,-\}} p(\mathbf{h}|t),$$

so

$$p(\pm|\mathbf{h}) = \frac{\mathcal{N}(\mathbf{m}^{\pm}, \Sigma^{\pm})[\mathbf{h}]}{\sum_{t\in\{+,-\}} \mathcal{N}(\mathbf{m}^{t}, \Sigma^{t})[\mathbf{h}]}.$$

Furthermore, it is easy to show that under the normality assumption, the resulting classifier is optimal in the Bayesian sense.

Remark 3 If data in feature space come from Normal distributions $\mathcal{N}(\mathbf{m}^{\pm}, \Sigma^{\pm})$ then $\boldsymbol{\beta}$ given by EEM minimizes probability of misclassification. More strictly speaking, if we draw $\mathbf{h}^{+}$ with probability 1 / 2 from $\mathcal{N}(\mathbf{m}^{+}, \Sigma^{+})$ and $\mathbf{h}^{-}$ with 1/2 from $\mathcal{N}(\mathbf{m}^{-}, \Sigma^{-})$ then for any $\boldsymbol{\alpha} \in \mathbb{R}^{h}$

$$p(\mp|\boldsymbol{\beta}^{\mathrm{T}}\mathbf{h}^{\pm}) \leq p(\mp|\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{h}^{\pm}).$$

5 Theory: learning capabilities

First, we show that under some simplifying assumptions, proposed method behaves as Extreme Learning Machine (or Weighted Extreme Learning Machine [30]).

Before proceeding further, we would like to remark that there are two popular notations for projecting data onto hyperplanes. One, used in ELM model, assume that $\mathbf{H}$ is a row matrix and $\boldsymbol{\beta}$ is a column vector, which results in the projection's equation $\mathbf{H}\boldsymbol{\beta}$. Second one, used in SVM and in our paper, assumes that both $\mathbf{H}$ and $\boldsymbol{\beta}$ are column oriented, which results in the $\beta^{\mathrm{T}}\mathbf{H}$ projection. In the following theorem, we will show some duality between $\boldsymbol{\beta}$ found by ELM and by EEM. To do so, we will need to change the notation during the proof, which will be indicated.

Theorem 1 *Let us assume that we are given an arbitrary, balanced binary dataset which can be perfectly learned by ELM with N hidden neurons. If this dataset points' image through random neurons* $\mathbf{H} = \varphi(\mathbf{X})$ *is centered (points' images have* 0 *mean) and classes have homogeneous covariances (we assume that there exist real* a_+ *and* a_- *such that* $\mathrm{cov}(\mathbf{H}) = a_+\mathrm{cov}(\mathbf{H}^{+}) = a_-\mathrm{cov}(\mathbf{H}^{-})$*) then EEM with the same hidden layer will also learn this dataset perfectly (with* 0 *error).*

Proof In the first part of the proof, we use the ELM notation. Projected data are centered, so $\mathrm{cov}(\mathbf{H}) = \mathbf{H}^{\mathrm{T}}\mathbf{H}$. ELM is able to learn this dataset perfectly, consequently $\mathbf{H}$ is invertible, thus also $\mathbf{H}^{\mathrm{T}}\mathbf{H}$ is invertible, as a result $\mathrm{cov}_{\dagger}(\mathbf{H}) = \mathrm{cov}(\mathbf{H}) = \mathbf{H}^{\mathrm{T}}\mathbf{H}$. We will now show that $\exists_{a\in\mathbb{R}_+}\boldsymbol{\beta}_{\mathrm{ELM}} = a\cdot\boldsymbol{\beta}_{\mathrm{EEM}}$. First, let us recall that $\boldsymbol{\beta}_{\mathrm{ELM}} = \mathbf{H}^{\dagger}\mathbf{t} = \mathbf{H}^{-1}\mathbf{t}$ and $\boldsymbol{\beta}_{\mathrm{EEM}} = \frac{2(\Sigma^{+}+\Sigma^{-})^{-1}(\mathbf{m}^{+}-\mathbf{m}^{-})}{\|\mathbf{m}^{+}-\mathbf{m}^{-}\|^2_{\Sigma^{-}+\Sigma^{+}}}$ where $\Sigma^{\pm} = \mathrm{cov}_{\dagger}(\mathbf{H}^{\pm})$. Due to the assumption of geometric homogeneity $\boldsymbol{\beta}_{\mathrm{EEM}} = \frac{2}{\|\mathbf{m}^{+}-\mathbf{m}^{-}\|^2_{\Sigma}}(\frac{a_+ + a_-}{a_+ a_-}\Sigma)^{-1}(\mathbf{m}^{+}-\mathbf{m}^{-})$, where $\Sigma = \mathrm{cov}_{\dagger}(\mathbf{H})$. Therefore,

$$\begin{aligned}\boldsymbol{\beta}_{\mathrm{ELM}} &= \mathbf{H}^{-1}\mathbf{t}\\ &= (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{t}\\ &= \mathrm{cov}_{\dagger}{}^{-1}(\mathbf{H})\mathbf{H}^{\mathrm{T}}\mathbf{t}.\end{aligned}$$

From now, we change the notation back to the one used in this paper and obtain

$$\begin{aligned}\boldsymbol{\beta}_{\mathrm{ELM}} &= \Sigma^{-1}\left(\sum_{\mathbf{h}^{+}\in\mathbf{H}^{+}}(+1\cdot\mathbf{h}^{+}) + \sum_{\mathbf{h}^{-}\in\mathbf{H}^{-}}(-1\cdot\mathbf{h}^{-})\right)\\ &= \Sigma^{-1}\left(\sum_{\mathbf{h}^{+}\in\mathbf{H}^{+}}\mathbf{h}^{+} - \sum_{\mathbf{h}^{-}\in\mathbf{H}^{-}}\mathbf{h}^{-}\right)\\ &= \Sigma^{-1}\frac{N}{2}(\mathbf{m}^{+}-\mathbf{m}^{-})\\ &= \frac{N}{2}\frac{\|\mathbf{m}^{+}-\mathbf{m}^{-}\|^2_{\Sigma}}{2}\frac{a_+ + a_-}{a_+ a_-}\boldsymbol{\beta}_{\mathrm{EEM}}\\ &= a\cdot\boldsymbol{\beta}_{\mathrm{EEM}},\end{aligned}$$

for $a = \frac{N}{2}\frac{\|\mathbf{m}^{+}-\mathbf{m}^{-}\|^2_{\Sigma}}{2}\frac{a_+ + a_-}{a_+ a_-} \in \mathbb{R}_+$. Again from homogeneity we obtain just one equilibrium point, located in the $\boldsymbol{\beta}^{\mathrm{T}}_{\mathrm{EEM}}(\mathbf{m}^{+}-\mathbf{m}^{-})/2$which results in the exact same classifier as the one given by ELM. This completes the proof. □

Similar result holds for EEKM and Least Squares Support Vector Machine.

Theorem 2 *Let us assume that we are given arbitrary, balanced binary dataset which can be perfectly learned by LS-SVM. If dataset points' images through Kernel-induced*

projection φ_{K} have homogeneous classes' covariances (*we assume that there exist real a_+ and a_- such that* $\mathrm{cov}(\varphi_{\mathrm{K}}(\mathbf{X})) = a_+\mathrm{cov}(\varphi_{\mathrm{K}}(\mathbf{X}^+)) = a_-\mathrm{cov}(\varphi_{\mathrm{K}}(\mathbf{X}^-)))$ *then EEKM with the same kernel and N hidden neurons will also learn this dataset perfectly* (*with* 0 *error*).

Proof It is a direct consequence of the fact that with N hidden neurons and homogeneous classes projections covariances, EEKM degenerates to the kernelized Fisher Discriminant which, as Gestel et al. showed [28], is equivalent to the solution of the Least Squares SVM. □

Both theorems can be extended to non-balanced datasets if we consider a Weighted ELM and Balanced LS-SVM. Proposed method has a balanced nature, so it internally assumes that classes priors are equal to 1/2. In the proofs, we show that when this is a true assumption, ELM and LS-SVM lead (under some assumptions) to the same solution. If one includes the same assumption in these two methods (through Weighted ELM and Balanced LS-SVM), then they again will solve the same problem despite true classes priors. We omit the exact proof as they are analogous to the above.

6 Practical considerations

In previous sections, we investigated the limiting case when $\dim(\mathcal{H}) = \infty$. However, in practice, we choose h random nonlinear projections which embed data in a high-dimensional space (with dimension at least h, but we can still consider it as an image in higher dimensional space, analogously to how Gaussian kernel actually maps to infinitely dimensional space by projecting through just N functions). As we show in further evaluation, it is sufficient to use h which is much smaller than N, so resulting computational complexities, cubic in h, are acceptable.

We can formulate the whole EEM training as a very simple algorithm (see Algorithms 1, 2).

Resulting model consists of three elements:

- feature projection function φ,
- linear operator $\boldsymbol{\beta}$,
- classification rule F.

As described before, F can be further compressed to just one or two thresholds $t_\pm$ using equations from previous sections. Either way, complexity of the resulting model is linear in terms of hidden units and classification of the new point takes $\mathcal{O}(dh)$ time.

During EEM training, the most expensive part of the algorithm is the computation of the covariance estimators and inversion of the sum of covariances. Even computation of the empirical covariance takes $\mathcal{O}(Nh^2)$ time so the total complexity of training, equal to $\mathcal{O}(h^3 + Nh^2) = \mathcal{O}(Nh^2)$, is acceptable. It is worth noting that training of the ELM also takes exactly $\mathcal{O}(Nh^2)$ time as it requires computation of $\mathbf{H}^{\mathrm{T}}\mathbf{H}$ for $\mathbf{H} \in \mathbb{R}^{N\times h}$. Training of EEMK requires additional computation of the square root of the sampled kernel matrix inverse $\mathrm{K}(\mathbf{X}^{[h]}, \mathbf{X}^{[h]})^{-1/2}$ but as $\mathrm{K}(\mathbf{X}^{[h]}, \mathbf{X}^{[h]}) \in \mathbb{R}^{h\times h}$ can be computed in $\mathcal{O}(dh^2)$ and both inverting and square rooting can be done in $\mathcal{O}(h^3)$ we obtain exact same asymptotical computational complexity as the one of EEM. Procedure of square rooting and inverting are both always possible as assuming that K is a valid kernel in the Mercer's sense yields that $\mathrm{K}(\mathbf{X}^{[h]}, \mathbf{X}^{[h]})$ is strictly positive definite and thus invertible. Further comparison of EEM, ELM and SVM is summarized in Table 2.

Next aspect we would like to discuss is the cost-sensitive learning. EEMs are balanced models in the sense that they are trying to maximize the balanced quality measures (like Balanced Accuracy or GMean). However, in practical applications, it might be the case that we are actually more interested in the positive class than the negative one (like in the medical applications). Proposed model gives a direct probability estimates of $p(\boldsymbol{\beta}^{\mathrm{T}}\mathbf{h}|t)$, which we can easily convert to the cost-sensitive classifier by introducing the prior probabilities of each class. Directly from Bayes

Algorithm 1 Extreme Entropy (Kernel) Machine

train$(\mathbf{X}^+, \mathbf{X}^-)$
  build φ **using Algorithm 2**
  $\mathbf{H}^{\pm} \leftarrow \varphi(\mathbf{X}^{\pm})$
  $\mathbf{m}^{\pm} \leftarrow 1/|\mathbf{H}^{\pm}| \sum_{\mathbf{h}^{\pm}\in\mathbf{H}^{\pm}} \mathbf{h}^{\pm}$
  $\boldsymbol{\Sigma}^{\pm} \leftarrow \mathrm{cov}_{\dagger}(\mathbf{H}^{\pm})$
  $\boldsymbol{\beta} \leftarrow 2\left(\boldsymbol{\Sigma}^+ + \boldsymbol{\Sigma}^-\right)^{-1}(\mathbf{m}^+ - \mathbf{m}^-)/\|\mathbf{m}^+ - \mathbf{m}^-\|_{\boldsymbol{\Sigma}^+ + \boldsymbol{\Sigma}^-}$
  $\mathrm{F}(x) = \arg\max_{t\in\{+,-\}} \mathcal{N}(\boldsymbol{\beta}^T\mathbf{m}^t, \boldsymbol{\beta}^T\boldsymbol{\Sigma}^t\boldsymbol{\beta})[x]$
  return $\boldsymbol{\beta}, \varphi, \mathrm{F}$

predict$(\mathbf{X})$
  return $\mathrm{F}(\boldsymbol{\beta}^T\varphi(\mathbf{X}))$

Algorithm 2 φ building

Extreme Entropy Machine(G, h)
select randomly $\mathbf{w}_i, b_i$ **for** $i \in \{1, ..., h\}$
$\varphi(\mathbf{x}) = [\mathrm{G}(\mathbf{x}, \mathbf{w}_1, b_1), ..., \mathrm{G}(\mathbf{x}, \mathbf{w}_h, b_h)]^T$
return φ

Extreme Entropy Kernel Machine$(\mathrm{K}, h, \mathbf{X})$
select randomly $\mathbf{X}^{[h]} \subset \mathbf{X}, |\mathbf{X}^{[h]}| = h$
$\mathrm{K}^{[h]} \leftarrow \mathrm{K}(\mathbf{X}^{[h]}, \mathbf{X}^{[h]})^{-1/2}$
$\varphi_{\mathrm{K}}(\mathbf{x}) = \mathrm{K}^{[h]}\mathrm{K}(\mathbf{X}^{[h]}, \mathbf{x})$
return φ_{K}

Table 2 Comparison of considered classifiers

	ELM	SVM	LS-SVM	EE(K)M
Optimization method	Linear regression	Quadratic programming	Linear system	Fisher discriminant
Nonlinearity	Random projection	Kernel	Kernel	Random (kernel) projection
Closed-form	Yes	No	Yes	Yes
Balanced	No[a]	No[a]	No[a]	Yes
Regression	Yes	No[a]	Yes	No
Criterion	Mean squared error	Hinge loss	Mean squared error	Entropy optimization
Learning theory	Huang et al. [11]	Vapnik et al. [4]	Suykens et al. [23]	This paper
No. of thresholds	1	1	1	1 or 2
Problem type	Regression	Classification	Regression	Classification
Model learning	Discriminative	Discriminative	Discriminative	Generative
Direct probability estimates	No	No	No	Yes
Training complexity	$\mathcal{O}(Nh^2)$	$\mathcal{O}(N^3)$	$\mathcal{O}(N^{2.34})$	$\mathcal{O}(Nh^2)$
Resulting model complexity	hd	$\|SV\|d$, $\|SV\| \ll N$	$Nd + 1$	$hd + 4$
Memory requirements	$\mathcal{O}(Nd)$	$\mathcal{O}(Nd)$	$\mathcal{O}(N^2)$	$\mathcal{O}(Nd)$
Source of regularization	Moore–Penrose pseudoinverse	Margin maximization	Quadratic loss penalty term	Ledoit–Wolf estimator
Hyperparameters	h, G	C, K	C, K	h, G or h, K
Number of classes	Any	2	2	2

By $|SV|$ we denote number of support vectors

[a] Features which can be added to a particular model by some minor modifications

Theorem, given $p(+)$ and $p(-)$, we can label our new sample $\mathbf{h}$ according to

$$p(t|\boldsymbol{\beta}^{\mathrm{T}}\mathbf{h}) \propto p(t)p(\boldsymbol{\beta}^{\mathrm{T}}\mathbf{h}|t),$$

so if we are given costs $C_+, C_- \in \mathbb{R}_+$ we can use them as weighting of priors

$$cl(\mathbf{x}) = \underset{t\in\{-,+\}}{\operatorname{argmax}} \frac{C_t}{C_- + C_+} p(\boldsymbol{\beta}^{\mathrm{T}}\mathbf{h}|t).$$

Let us now investigate the possible efficiency bottleneck. In EEKM, the classification of the new point $\mathbf{h}$ is based on

$$\begin{aligned} cl(\mathbf{x}) &= \mathrm{F}(\boldsymbol{\beta}^{\mathrm{T}}\varphi_{\mathrm{K}}(\mathbf{x})) \\ &= \mathrm{F}(\boldsymbol{\beta}^{\mathrm{T}}(\mathrm{K}(\mathbf{x}, \mathbf{X}^{[h]})\mathrm{K}^{[h]})^{\mathrm{T}}) \\ &= \mathrm{F}(\boldsymbol{\beta}^{\mathrm{T}}(\mathrm{K}^{[h]})^{\mathrm{T}}\mathrm{K}(\mathbf{x}, \mathbf{X}^{[h]})^{\mathrm{T}}) \\ &= \mathrm{F}((\mathrm{K}^{[h]}\boldsymbol{\beta})^{\mathrm{T}}\mathrm{K}(\mathbf{X}^{[h]}, \mathbf{x})). \end{aligned}$$

One can convert EEKM to the SLFN by putting:

$$\begin{aligned} \hat{\varphi}_{\mathrm{K}}(\mathbf{x}) &= \mathrm{K}(\mathbf{X}^{[h]}, \mathbf{x}) \\ \hat{\boldsymbol{\beta}} &= \mathrm{K}^{[h]}\boldsymbol{\beta}, \end{aligned}$$

so the classification rule becomes

$$cl(\mathbf{x}) = \mathrm{F}(\hat{\boldsymbol{\beta}}^{\mathrm{T}} \hat{\varphi}_{\mathrm{K}}(\mathbf{x})).$$

This way complexity of the new point's classification is exactly the same as in the case of EEM and ELM (or any other SLFN).

7 Evaluation

For the evaluation purposes, we implemented five methods, namely: Weighted Extreme Learning Machine (WELM [30]), Extreme Entropy Machine (EEM), Extreme Entropy Kernel Machine (EEKM), Least Squares Support Vector Machines (LS-SVM [23]) and Support Vector Machines (SVM [4]). All experiments were performed using machine with Intel Xeon 2.8Ghz processors with enough RAM to fit any required computations.

All methods with the exception of SVM were implemented using Python with use of NUMPY [27] and SCIPY [14] libraries included in ANACONDA[4] for fair comparison. For SVM, we used highly efficient LIBSVM [3] library with bindings available in SCIKIT-LEARN [19]. Random projection-based methods (WELM, EEM) were tested using following three generalized activation functions $\mathrm{G}(\mathbf{x}, \mathbf{w}, b)$

- sigmoid (SIG): $\frac{1}{1+\exp(-\langle \mathbf{w},\mathbf{x}\rangle+b)}$,
- normalized sigmoid (NSIG): $\frac{1}{1+\exp(-\langle \mathbf{w},\mathbf{x}\rangle/d+b)}$,
- radial basis function (RBF): $\exp(-b\|\mathbf{w}-\mathbf{x}\|^2)$.

Random parameters (weights and biases) were selected from uniform distributions on [0, 1]. Training of WELM was performed using Moore–Penrose pseudoinverse and of EEM using Ledoit–Wolf covariance estimator, as both are parameterless, closed-form estimators of required objects. For kernel methods (EEKM, LS-SVM, SVM), we used the Gaussian kernel (RBF) $\mathrm{K}_\gamma(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2)$. In all methods requiring class balancing schemes (WELM, LS-SVM, SVM), we used balance weights w_i equal to the ratio of bigger class and current class (so $\sum_{i=1}^{N} w_i t_i = 0$, which is equivalent to having $w_i = (\max_{t\in\{-,+\}} N_t)/N_{t_i}$).

Hyperparameters of each model were fitted, performed grid search included: hidden layer size $h = 50, 100, 250, 500, 1000$ (WELM, EEM, EEKM), Gaussian Kernel width $\gamma = 10^{-10}, \ldots, 10^{0}$ (EEKM, LS-SVM, SVM), SVM regularization parameter $C = 10^{-1}, \ldots, 10^{10}$ (LS-SVM, SVM).

Datasets' features were linearly scaled (per feature) to have each feature in the interval [0, 1]. No other data whitening/filtering was performed. All experiments were conducted in repeated tenfold stratified cross-validation.

We use following evaluation metric

$$\mathrm{GMean}(\mathrm{TP}, \mathrm{FP}, \mathrm{TN}, \mathrm{FN}) = \sqrt{\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \cdot \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}},$$

where TP represents true positives, TN true negatives, FP false positives, FN false negatives. We choose it due to the balanced nature and usage in previous works regarding Weighted Extreme Learning Machines [30], however very similar results can be obtained for Balanced Accuracy (which is an arithmetic mean of accuracies over each class).

Table 3 Characteristics of used datasets

Dataset	d	$\lvert\mathbf{X}^-\rvert$	$\lvert\mathbf{X}^+\rvert$
AUSTRALIAN	14	383	307
BREAST CANCER	9	444	239
DIABETES	8	268	500
GERMAN NUMER	24	700	300
HEART	13	150	120
LIVER DISORDERS	6	145	200
SONAR	60	111	97
SPLICE	60	483	517
ABALONE7	10	3786	391
ARYTHMIA	261	427	25
CAR EVALUATION	21	1594	134
ECOLI	7	301	35
LIBRAS MOVE	90	336	24
OIL SPILL	48	896	41
SICK EUTHYROID	42	2870	293
SOLAR FLARE	32	1321	68
SPECTROMETER	93	486	45
FOREST COVER	54	571519	9493
ISOLET	617	7197	600
MAMMOGRAPHY	6	10923	260
PROTEIN HOMOLOGY	74	144455	1296
WEBPAGES	300	33799	981

7.1 Basic UCI datasets

We start our experiments with nine datasets coming from UCI REPOSITORY [2], namely AUSTRALIAN, BREAST-CANCER, DIABETES, GERMAN.NUMER, HEART, IONOSPHERE, LIVER DISORDERS, SONAR and SPLICE, summarized in Table 3. This dataset includes rather balanced, low-dimensional problems.

On such data, EEM seems to perform noticeably better than ELM when using RBF activation function (see

[4] https://store.continuum.io/cshop/anaconda/.

Table 4 GMean on all considered datasets

	$WELM_{sig}$	EEM_{sig}	$WELM_{nsig}$	EEM_{nsig}	$WELM_{rbf}$	EEM_{rbf}	$LS\text{-}SVM_{rbf}$	$EEKM_{rbf}$	SVM_{rbf}
AUSTRALIAN	86.3 ± 4.5	**87.0** ± 4.0	85.9 ± 4.4	86.5 ± 3.2	85.8 ± 4.9	86.9 ± 4.4	86.9 ± 4.1	86.8 ± 3.8	86.8 ± 3.7
BREAST-CANCER	96.9 ± 1.7	97.3 ± 1.2	97.6 ± 1.5	97.4 ± 1.2	96.6 ± 1.8	97.3 ± 1.1	97.6 ± 1.3	**97.8** ± 1.1	96.8 ± 1.7
DIABETES	74.2 ± 4.6	74.5 ± 4.6	74.1 ± 5.5	74.9 ± 5.0	73.2 ± 5.6	74.9 ± 5.9	75.5 ± 5.6	**75.7** ± 5.6	74.8 ± 3.5
GERMAN	68.8 ± 6.9	71.3 ± 4.1	70.7 ± 6.1	72.4 ± 5.4	71.1 ± 6.1	72.2 ± 5.7	73.2 ± 4.5	72.9 ± 5.3	**73.4** ± 5.4
HEART	78.8 ± 6.3	82.5 ± 7.4	78.1 ± 7.0	83.7 ± 7.2	80.2 ± 8.9	81.9 ± 6.9	83.7 ± 8.5	83.6 ± 7.5	**84.6** ± 7.0
IONOSPHERE	71.5 ± 9.5	77.0 ± 12.8	82.7 ± 7.8	84.6 ± 9.1	85.6 ± 8.4	90.8 ± 5.2	91.2 ± 5.5	93.4 ± 4.3	**94.7** ± 3.9
LIVER DISORDERS	68.1 ± 8.0	68.6 ± 8.9	66.3 ± 8.2	62.1 ± 8.1	67.2 ± 5.9	71.4 ± 7.0	71.1 ± 8.3	70.2 ± 6.9	**72.3** ± 6.2
SONAR	66.7 ± 10.1	70.1 ± 11.5	80.2 ± 7.4	78.3 ± 11.2	83.2 ± 6.9	82.8 ± 5.2	86.5 ± 5.4	**87.0** ± 7.5	83.0 ± 7.1
SPLICE	64.7 ± 2.8	49.4 ± 5.5	81.8 ± 3.2	80.9 ± 2.7	75.5 ± 3.9	82.2 ± 3.5	**89.9** ± 3.0	88.0 ± 4.0	88.0 ± 2.2
ABALONE7	79.7 ± 2.3	79.8 ± 3.5	80.0 ± 2.8	76.1 ± 3.7	80.1 ± 3.2	79.7 ± 3.6	**80.2** ± 3.4	79.9 ± 3.4	79.7 ± 2.7
ARYTHMIA	28.3 ± 35.4	40.3 ± 20.9	64.2 ± 24.6	**85.6** ± 10.3	66.9 ± 25.8	79.4 ± 12.5	84.4 ± 10.0	85.2 ± 10.6	80.9 ± 11.8
CAR EVALUATION	99.1 ± 0.3	98.9 ± 0.4	99.0 ± 0.3	97.9 ± 0.6	99.0 ± 0.3	98.5 ± 0.3	99.5 ± 0.2	99.2 ± 0.3	**100.0** ± 0.0
ECOLI	86.9 ± 6.5	88.3 ± 7.1	86.9 ± 6.8	88.6 ± 6.9	86.4 ± 7.0	88.8 ± 7.2	89.2 ± 6.3	**89.4** ± 6.9	88.5 ± 6.2
LIBRAS MOVE	65.5 ± 10.7	19.3 ± 8.1	82.5 ± 12.0	93.0 ± 11.8	89.6 ± 11.9	93.9 ± 11.9	96.5 ± 8.6	**96.6** ± 8.7	91.6 ± 11.9
OIL SPILL	86.0 ± 6.9	**88.8** ± 6.5	83.8 ± 7.6	84.7 ± 8.7	85.8 ± 9.3	88.1 ± 6.1	86.7 ± 8.4	87.2 ± 4.9	85.7 ± 11.4
SICK EUTHYROID	88.1 ± 1.7	87.9 ± 2.4	88.5 ± 2.1	81.7 ± 2.7	89.1 ± 1.9	88.2 ± 2.4	89.5 ± 1.7	89.3 ± 1.9	**90.9** ± 2.0
SOLAR FLARE	60.4 ± 16.8	63.7 ± 12.9	61.3 ± 10.8	67.4 ± 9.0	60.3 ± 14.8	68.9 ± 9.3	67.3 ± 8.8	67.3 ± 9.0	**70.9** ± 8.5
SPECTROMETER	82.9 ± 13.0	87.3 ± 7.8	88.0 ± 10.8	90.2 ± 8.6	86.6 ± 8.2	93.0 ± 14.6	94.6 ± 8.4	93.5 ± 14.7	**95.4** ± 5.1
FOREST COVER	90.8 ± 0.3	90.5 ± 0.3	90.7 ± 0.3	85.1 ± 0.4	90.9 ± 0.3	87.1 ± 0.0	–	**91.8** ± 0.3	–
ISOLET	0.0 ± 0.0	0.0 ± 0.0	96.3 ± 0.7	95.6 ± 1.1	93.0 ± 0.9	91.4 ± 1.0	**98.0** ± 0.7	97.4 ± 0.6	97.6 ± 0.6
MAMMOGRAPHY	90.4 ± 2.8	89.0 ± 3.2	90.7 ± 3.3	87.2 ± 3.0	89.9 ± 3.8	89.5 ± 3.1	**91.0** ± 3.1	89.5 ± 3.1	89.8 ± 3.8
PROTEIN HOMOLOGY	95.3 ± 0.8	94.9 ± 0.8	95.1 ± 0.9	94.2 ± 1.3	95.0 ± 1.0	95.1 ± 1.1	–	**95.7** ± 0.9	–
WEBPAGES	72.0 ± 0.0	73.1 ± 2.0	93.0 ± 1.8	**93.1** ± 1.7	86.7 ± 0.0	84.4 ± 1.6	–	**93.1** ± 1.7	**93.1** ± 1.7

Table 4), and rather similar when using sigmoid one—in such a scenario, for some datasets, ELM achieves better results while for other EEM wins. Results obtained for EEKM are comparable with those obtained by LS-SVM and SVM, in both cases proposed method achieves better results on about third of problems, on the third it draws and on a third it loses. This experiment can be seen as a proof of concept of the whole methodology, showing that it can be truly a reasonable alternative for existing models in some problems. It appears that contrary to ELM, proposed methods (EEM and EEKM) achieve best scores across all considered models in some of the datasets regardless of the used activation function/kernel (only Support Vector Machines and their least squares counterpart are competitive in this sense).

7.2 Highly unbalanced datasets

In the second part, we considered the nine highly unbalanced datasets, summarized in the second part of Table 3. Ratio between bigger and smaller class varies from 10:1 to even 20:1 which makes them really hard for unbalanced models. Obtained results (see Table 4) resemble those obtained on UCI repository. We can see better results in about half of experiments if we fix a particular activation function/kernel (so we compare ELM_x with EEM_x and $LS\text{-}SVM_x$ with $EEKM_x$).

Table 5 shows that training time of Extreme Entropy Machines is comparable with the ones obtained by Extreme Learning Machines (differences on the level of 0.1–0.2 are not significant on such datasets' sizes). We have a robust method which learns in below two seconds a model for hundreds/thousands of examples. For larger datasets (like ABALONE7 or SICK EUTHYROID) proposed methods not only outperform SVM and LS-SVM in terms of robustness but there is also noticeable difference between their training times and ELMs. This suggests that even though ELM and EEM are quite similar and on small datasets are equally fast, EEM can better scale up to truly big datasets. Obviously obtained training times do not resemble the full training time as it strongly depends on the technique used for hyperparameters selection and resolution of grid search (or other parameters tuning technique). In such full scenario, training times of SVM-related models are

Table 5 Highly unbalanced datasets times in seconds using machine with Intel Xeon 2.8 GHz processors

	$WELM_{sig}$	EEM_{sig}	$WELM_{nsig}$	EEM_{nsig}	$WELM_{rbf}$	EEM_{rbf}	LS-SVM_{rbf}	$EEKM_{rbf}$	SVM_{rbf}
ABALONE7	1.9	1.2	2.5	1.6	1.8	**1.2**	20.8	1.9	4.7
ARYTHMIA	0.2	0.7	0.3	0.9	0.3	0.7	**0.1**	0.3	0.1
CAR EVALUATION	1.3	0.9	1.5	1.0	1.1	0.9	2.0	1.4	**0.1**
ECOLI	0.2	0.8	0.2	0.8	0.1	0.7	**0.0**	0.1	0.2
LIBRAS MOVE	0.2	0.9	0.2	0.8	0.1	0.7	0.0	0.1	**0.0**
OIL SPILL	0.7	0.8	0.6	0.8	0.6	0.8	0.4	0.9	**0.1**
SICK EUTHYROID	1.5	1.1	1.4	**1.1**	1.5	1.1	9.6	1.7	21.0
SOLAR FLARE	**0.7**	0.8	0.7	0.8	0.8	0.8	1.1	1.3	16.1
SPECTROMETER	0.2	0.7	0.3	0.7	0.2	0.7	0.1	0.3	**0.0**
FOREST COVER	110.7	104.6	144.9	45.6	111.3	**38.2**	> 600	107.4	> 600
ISOLET	9.7	4.5	4.9	3.0	3.4	**2.1**	126.9	3.2	53.5
MAMMOGRAPHY	4.0	**2.2**	6.1	3.0	4.0	2.2	327.3	3.3	9.5
PROTEIN HOMOLOGY	27.6	**21.6**	86.3	27.9	62.5	22.0	> 600	30.7	> 600
WEBPAGES	16.0	**6.2**	14.5	8.5	7.1	6.4	> 600	9.0	217.0

significantly bigger due to the requirement of exact tuning of both C and γ in real domains.

7.3 Extremely unbalanced datasets

Third part of experiments involves analysis of extremely unbalanced datasets (with class imbalance up to 100:1) containing tens and hundreds thousands of examples. Five analyzed datasets span from NLP tasks (WEBPAGES) through medical applications (MAMMOGRAPHY) to bioinformatics (PROTEIN HOMOLOGY). This type of dataset often occurs in the true data mining which makes these results much more practical than the one obtained on small/balanced data. Hyperparameters of each method are carefully fitted as described in the previous section.

Scores obtained on ISOLET dataset (see Table 4) for sigmoid-based random projections are a result of very high values ($\approx$200) of $\langle \mathbf{x}, \mathbf{w} \rangle$ for all $\mathbf{x}$, which results in $\mathrm{G}(\mathbf{x}, \mathbf{w}, b) = 1$, so the whole dataset is reduced to the singleton $\{[1, \ldots, 1]^{\mathrm{T}}\} \subset \mathbb{R}^h \subset \mathcal{H}$which obviously is not separable by any classifier, neither ELM nor EEM.

For other activation functions, we see that EEM achieves slightly worse results than ELM. On the other hand, scores of EEKM generally outperform the ones obtained by ELM and are very close to the ones obtained by well-tuned SVM and LS-SVM. In the same time, EEM and EEKM were trained significantly faster, as Table 5 shows, it was order of magnitude faster than SVM-related models and even $1.5 - 2\times$ faster than ELM. It seems that the Ledoit–Wolf covariance estimation computation with this matrices inversion is simply a faster operation (scales better) than computation of the Moore–Penrose pseudoinverse of the $\mathbf{H}^{\mathrm{T}}\mathbf{H}$. Obviously, one can alternate ELM training routine to the regularized one where instead of $(\mathbf{H}^{\mathrm{T}}\mathbf{H})^{\dagger}$ one computes

$$(\mathbf{H}^{\mathrm{T}}\mathbf{H} + \mathbf{I}/C)^{-1}, \tag{5}$$

but we are analyzing here approach without parametrized regularization, while the analog could be used for EEM in the form of

$$(\mathrm{cov}(\mathbf{X}^{-}) + \mathrm{cov}(\mathbf{X}^{+}) + \mathbf{I}/C)^{-1} \tag{6}$$

instead of computing Ledoit–Wolf estimator. In other words, in the regularization parameterless training scenario, as described in this paper, EEMs seem to scale better than ELMs while still obtaining similar classification results. In the same time, EEKM obtains SVM-level results with orders of magnitude faster training. Both ELM and EEM could be transformed into regularization parameter-based learning [see Eqs. (5), (6)], but this is beyond the scope of this work.

7.4 Entropy-based hyperparameters optimization

Now, we proceed to entropy-based evaluation. Given particular set of linear hypotheses $\mathcal{M}$ in $\mathcal{H}$, we want to select optimal set of hyperparameters θ (such as number of hidden neurons or regularization parameter) which identify a particular model $\boldsymbol{\beta}_{\theta} \in \mathcal{M} \subset \mathcal{H}$. Instead of using expensive internal cross-validation (or other generalization error estimation technique like $\mathrm{Err}^{0.632}$), we select such θwhich maximizes our entropic measure. In particular, we consider a simplified Cauchy–Schwarz Divergence-based strategy where we select θ maximizing

Table 6 UCI datasets GMean with parameters tuning based on selecting a model according to (a) $D_{CS}(\mathcal{N}(\boldsymbol{\beta}^T\mathbf{m}^+, \boldsymbol{\beta}^T\Sigma^+\boldsymbol{\beta}), \mathcal{N}(\boldsymbol{\beta}^T\mathbf{m}^-, \boldsymbol{\beta}^T\Sigma^-\boldsymbol{\beta}))$ and (b) $D_{CS}([\![\boldsymbol{\beta}^T\mathbf{h}^+]\!], [\![\boldsymbol{\beta}^T\mathbf{h}^-]\!])$where $\boldsymbol{\beta}$ is a linear operator found by a particular optimization procedure instead of internal cross-validation

	$WELM_{sig}$	EEM_{sig}	$WELM_{nsig}$	EEM_{nsig}	$WELM_{rbf}$	EEM_{rbf}	LS-SVM_{rbf}	$EEKM_{rbf}$	SVM_{rbf}
(a) $D_{CS}(\mathcal{N}(\boldsymbol{\beta}^T\mathbf{m}^+, \boldsymbol{\beta}^T\Sigma^+\boldsymbol{\beta}), \mathcal{N}(\boldsymbol{\beta}^T\mathbf{m}^-, \boldsymbol{\beta}^T\Sigma^-\boldsymbol{\beta}))$									
AUSTRALIAN	51.2 ± 7.5	86.3 ± 4.8	50.3 ± 6.4	**86.5** ± 3.2	50.3 ± 8.5	86.2 ± 5.3	58.5 ± 7.9	85.2 ± 5.6	85.7 ± 4.7
BREAST-CANCER	83.0 ± 4.3	97.0 ± 1.6	72.0 ± 6.6	97.1 ± 1.9	77.3 ± 5.3	97.3 ± 1.1	79.2 ± 7.7	96.9 ± 1.4	**97.5** ± 1.2
DIABETES	52.3 ± 4.7	74.4 ± 4.0	51.7 ± 4.0	**74.7** ± 5.2	52.1 ± 3.7	73.5 ± 5.9	60.1 ± 4.2	72.2 ± 5.4	73.2 ± 5.9
GERMAN	57.1 ± 4.0	69.3 ± 5.0	51.7 ± 3.0	**72.4** ± 5.4	52.8 ± 6.3	70.9 ± 6.9	55.0 ± 4.3	67.8 ± 5.7	60.5 ± 4.5
HEART	68.6 ± 5.8	79.4 ± 6.9	65.6 ± 5.9	**82.9** ± 7.4	60.3 ± 9.4	77.4 ± 7.2	66.2 ± 4.2	77.7 ± 7.0	76.5 ± 6.6
IONOSPHERE	62.7 ± 10.6	77.0 ± 12.8	68.5 ± 5.1	84.6 ± 9.1	69.5 ± 9.6	90.8 ± 5.2	72.8 ± 6.1	93.4 ± 4.2	**94.7** ± 3.9
LIVER DISORDERS	53.2 ± 7.0	68.5 ± 6.7	52.2 ± 11.8	62.1 ± 8.1	53.9 ± 8.0	**71.4** ± 7.0	62.9 ± 7.8	69.6 ± 8.2	66.9 ± 8.0
SONAR	66.3 ± 6.1	66.1 ± 15.0	80.2 ± 7.4	76.9 ± 5.2	83.2 ± 6.9	82.8 ± 5.2	85.9 ± 4.9	**87.7** ± 6.1	86.6 ± 3.3
SPLICE	51.8 ± 4.3	49.4 ± 5.5	64.9 ± 3.1	80.2 ± 2.6	60.8 ± 3.5	82.2 ± 3.5	**89.7** ± 3.3	88.0 ± 4.0	89.5 ± 2.9
(b) $D_{CS}([\![\boldsymbol{\beta}^T\mathbf{h}^+]\!], [\![\boldsymbol{\beta}^T\mathbf{h}^-]\!])$									
AUSTRALIAN	51.2 ± 7.5	86.3 ± 4.8	50.3 ± 6.4	**86.5** ± 3.2	50.3 ± 8.5	86.2 ± 5.3	58.5 ± 7.9	85.2 ± 5.6	84.2 ± 4.1
BREAST-CANCER	83.0 ± 4.3	97.0 ± 1.6	72.0 ± 6.6	**97.4** ± 1.2	77.3 ± 5.3	97.3 ± 1.1	79.3 ± 7.1	96.9 ± 1.4	96.3 ± 2.4
DIABETES	52.3 ± 4.7	74.4 ± 4.0	51.7 ± 4.0	**74.7** ± 5.2	52.1 ± 3.7	73.5 ± 5.9	60.1 ± 4.2	72.2 ± 5.4	71.9 ± 5.4
GERMAN	57.1 ± 4.0	69.3 ± 5.0	51.7 ± 3.0	**71.7** ± 5.9	52.8 ± 6.3	70.9 ± 6.9	54.4 ± 5.7	67.8 ± 5.7	59.5 ± 4.2
HEART	60.0 ± 9.2	79.4 ± 6.9	65.6 ± 5.9	**82.9** ± 7.4	52.6 ± 9.0	77.4 ± 7.2	61.9 ± 5.8	77.7 ± 7.0	76.3 ± 7.7
IONOSPHERE	62.4 ± 8.1	77.0 ± 12.8	68.5 ± 5.1	84.6 ± 9.1	67.6 ± 9.8	90.8 ± 5.2	67.0 ± 10.7	**93.4** ± 4.2	92.3 ± 4.6
LIVER DISORDERS	50.9 ± 11.5	68.5 ± 6.7	50.4 ± 9.2	62.1 ± 8.1	53.9 ± 8.0	**71.4** ± 7.0	62.9 ± 7.8	69.6 ± 8.2	66.9 ± 8.0
SONAR	66.3 ± 6.1	66.1 ± 15.0	80.2 ± 7.4	76.9 ± 5.2	62.9 ± 9.4	82.8 ± 5.2	83.6 ± 4.5	**87.7** ± 6.1	86.6 ± 3.3
SPLICE	51.8 ± 4.3	33.1 ± 6.5	64.9 ± 3.1	80.2 ± 2.6	60.8 ± 3.5	82.2 ± 3.5	85.4 ± 4.1	88.0 ± 4.0	**89.5** ± 2.9

$$D_{CS}(\mathcal{N}(\boldsymbol{\beta}_\theta^T\mathbf{m}^+, \mathrm{var}(\boldsymbol{\beta}_\theta^T\mathbf{H}^+)), \mathcal{N}(\boldsymbol{\beta}_\theta^T\mathbf{m}^-, \mathrm{var}(\boldsymbol{\beta}_\theta^T\mathbf{H}^-))),$$

and kernel density-based entropic strategy [7] selecting θ maximizing

$$D_{CS}([\![\boldsymbol{\beta}_\theta^T\mathbf{H}^+]\!], [\![\boldsymbol{\beta}_\theta^T\mathbf{H}^-]\!]),$$

where $[\![A]\!] = [\![A]\!]_{\sigma(A)}$ is a Gaussian KDE using Silverman's rule of the window width [22]

$$\sigma(A) = \left(\frac{4}{3|A|}\right)^{1/5}\mathrm{std}(A) \approx \frac{1.06}{\sqrt[5]{|A|}}\mathrm{std}(A).$$

This way we can use whole given set for training and do not need to repeat the process, as D_{CS} is computed on the training set instead of the hold-out set.

First, one can notice on Table 6 that such entropic criterion works well for EEM, EEKM and Support Vector Machines. On the other hand, it is not very well suited for ELM models. This confirms conclusions from Czarnecki and Tabor work on classification using D_{CS} [7] where SVMs were claimed to be conceptually similar in terms of optimization objective, as well as widens it to the new class of models (EEMs). Second, Table 6 shows that EEM and EEKM can truly select their hyperparameters using very simple technique requiring no model retraining. Computation of

$$D_{CS}(\mathcal{N}(\boldsymbol{\beta}_\theta^T\mathbf{m}^+, \mathrm{var}(\boldsymbol{\beta}_\theta^T\mathbf{H}^+)), \mathcal{N}(\boldsymbol{\beta}_\theta^T\mathbf{m}^-, \mathrm{var}(\boldsymbol{\beta}_\theta^T\mathbf{H}^-)))$$

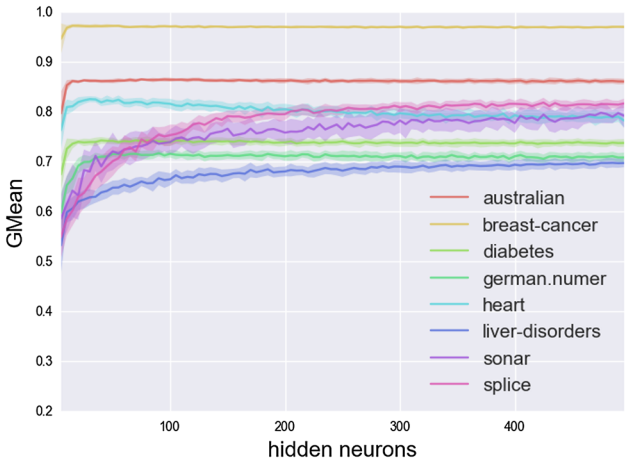

Fig. 3 *Plot* of the EEM's (with RBF activation function) GMean scores from cross-validation experiments for increasing sizes of hidden layer. *Error bars* denote standard deviation

is linear in terms of training set and constant time if performed using precomputed projections of required objects (which are either way computed during EEM training). This makes this very fast model even more robust.

7.5 EEM stability

It was previously reported [12] that ELMs have very stable results in the wide range of the number of hidden neurons.

We performed analogous experiments with EEM on UCI datasets. We trained models for 100 increasing hidden layers sizes ($h = 5, 10, \ldots, 500$) and plotted resulting GMean scores on Fig. 3.

One can notice that, similar to ELM, proposed methods are very stable. Once machine gets enough neurons (around 100 in case of tested datasets), further increasing of the feature space dimension has minor effect on the generalization capabilities of the model. It is also important that some of these datasets (like sonar) do not even have 500 points, so there are more dimensions in the Hilbert space than we have points to build our covariance estimates, and even though we still do not observe any rapid overfitting.

8 Conclusions

In this paper, we have presented Extreme Entropy Machines, models derived from the information theoretic measures and applied to the classification problems. Proposed methods are strongly related to the concepts of Extreme Learning Machines (in terms of general workflow, rapid training and randomization) as well as Support Vector Machines (in terms of margin maximization interpretation as well as LS-SVM duality).

Main characteristics of EEMs are:

- information theoretic background based on differential and Renyi's quadratic entropies,
- closed-form solution of the optimization problem,
- generative training, leading to direct probability estimates,
- small number of hyperparameters,
- good classification results,
- rapid training that scales well to hundreds of thousands of examples and beyond,
- theoretical and practical similarities to the large margin classifiers and Fisher Discriminant.

Performed evaluation showed that, similar to ELM, proposed EEM is a very stable model in terms of the size of the hidden layer and achieves comparable classification results to the ones obtained by SVMs and ELMs. Furthermore, we showed that our method scales better to truly big datasets (consisting of hundreds of thousands of examples) without sacrificing results quality.

During our considerations, we pointed out some open problems and issues, which are worth investigation:

- Can one construct a closed-form entropy-based classifier with different distribution families than Gaussians? It remains an open problem whether it is possible even for a convex combination of two Gaussians.
- Is there a theoretical justification of the stability of the extreme learning techniques? In particular, can one show whether performing random projection is equivalent to some prior on the decision function space like in the case of kernels?
- Is it possible to further increase achieved results by performing unsupervised entropy-based optimization in the hidden layer? For Gaussian nodes one could use some GMM clustering techniques, but is there an efficient way of selecting nodes with different activation functions, such as ReLU?

References

1. Anthony M (2003) Learning multivalued multithreshold functions. CDMA Research Report No. LSE-CDMA-2003-03, London School of Economics
2. Bache K, Lichman M (2013) UCI machine learning repository. http://archive.ics.uci.edu/ml. Accessed 30 June 2015
3. Chang CC, Lin CJ (2011) Libsvm: a library for support vector machines. ACM Trans Intell Syst Technol 2(3):27
4. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
5. Cover TM, Thomas JA (2012) Elements of information theory. Wiley, New York
6. Czarnecki WM, Tabor J (2014) Cluster based RBF kernel for support vector machines. ArXiv e-prints. http://arxiv.org/abs/1408.2869. Accessed 30 June 2015
7. Czarnecki WM, Tabor J (2014) Multithreshold Entropy Linear Classifier: Theory and applications. Expert Syst Appl 42(13):5591–5606
8. Dempster AP, Laird NM, Rubin DB Maximum likelihood from incomplete data via the em algorithm. In: Journal of the Royal Statistical Society. Series B (Methodological), JSTOR, pp 1–38 (1977)
9. Drineas P, Mahoney MW (2005) On the Nyström method for approximating a Gram matrix for improved kernel-based learning. J Mach Learn Res 6:2153–2175
10. Durrant RJ, Kaban A (2013) Sharp generalization error bounds for randomly-projected classifiers. Proceedings of International Conference on Machine Learning (ICML), pp 693–701
11. Huang GB, Zhu QY, Siew CK: Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of the 2004 IEEE international joint conference on neural networks, 2004, vol 2. IEEE, pp 985–990 (2004)
12. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1):489–501
13. Jenssen R, Principe JC, Erdogmus D, Eltoft T (2006) The Cauchy–Schwarz divergence and parzen windowing: connections to graph theory and mercer kernels. J Frankl Inst 343(6):614–629
14. Jones E, Oliphant T, Peterson P (2001) Scipy: open source scientific tools for python. http://www.scipy.org/. Accessed 30 June 2015

15. Kulkarni SR, Lugosi G, Venkatesh SS (1998) Learning pattern classification-a survey. IEEE Trans Inf Theory 44(6):2178–2206
16. Ledoit O, Wolf M (2004) A well-conditioned estimator for large-dimensional covariance matrices. J Multivar Anal 88(2):365–411
17. Mahalanobis PC (1936) On the generalized distance in statistics. Proc Natl Inst Sci (Calcutta) 2:49–55
18. Parzen E (1962) On estimation of a probability density function and mode. Ann Math Stat 33:1065–1076
19. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V et al (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12:2825–2830
20. Poggio T, Girosi F (1989) A theory of networks for approximation and learning. In: Tech. rep, DTIC document
21. Principe JC (2000) Information theoretic learning. Springer, Berlin
22. Silverman BW (1986) Density estimation for statistics and data analysis, vol 26. CRC Press, Boca Raton
23. Suykens JA, Vandewalle J (1999) Least squares support vector machine classifiers. Neural Process Lett 9(3):293–300
24. Suykens JA, De Brabanter J, Lukas L, Vandewalle J (2002) Weighted least squares support vector machines: robustness and sparse approximation. Neurocomputing 48(1):85–105
25. Tabor J, Spurek P (2014) Cross-entropy clustering. Pattern Recogn 47(9):3046–3059
26. Titterington DM, Smith AF, Makov UE et al (1985) Statistical analysis of finite mixture distributions, vol 7. Wiley, New York
27. Van Der Walt S, Colbert SC, Varoquaux G (2011) The numpy array: a structure for efficient numerical computation. Comput Sci Eng 13(2):22–30
28. Van Gestel T, Suykens JA, Baesens B, Viaene S, Vanthienen J, Dedene G, De Moor B, Vandewalle J (2004) Benchmarking least squares support vector machine classifiers. Mach Learn 54(1):5–32
29. Zhang, T., Zhou, Z.H.: Large margin distribution machine. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 313–322 (2014)
30. Zong W, Huang GB, Chen Y (2013) Weighted extreme learning machine for imbalance learning. Neurocomputing 101:229–242