Deep Learning

Lecture 2

Fernando Perez-Cruz
based on Thomas Hofmann lectures

Swiss Data Science Center
ETH Zurich and EPFL – `datascience.ch`

October 1st, 2018

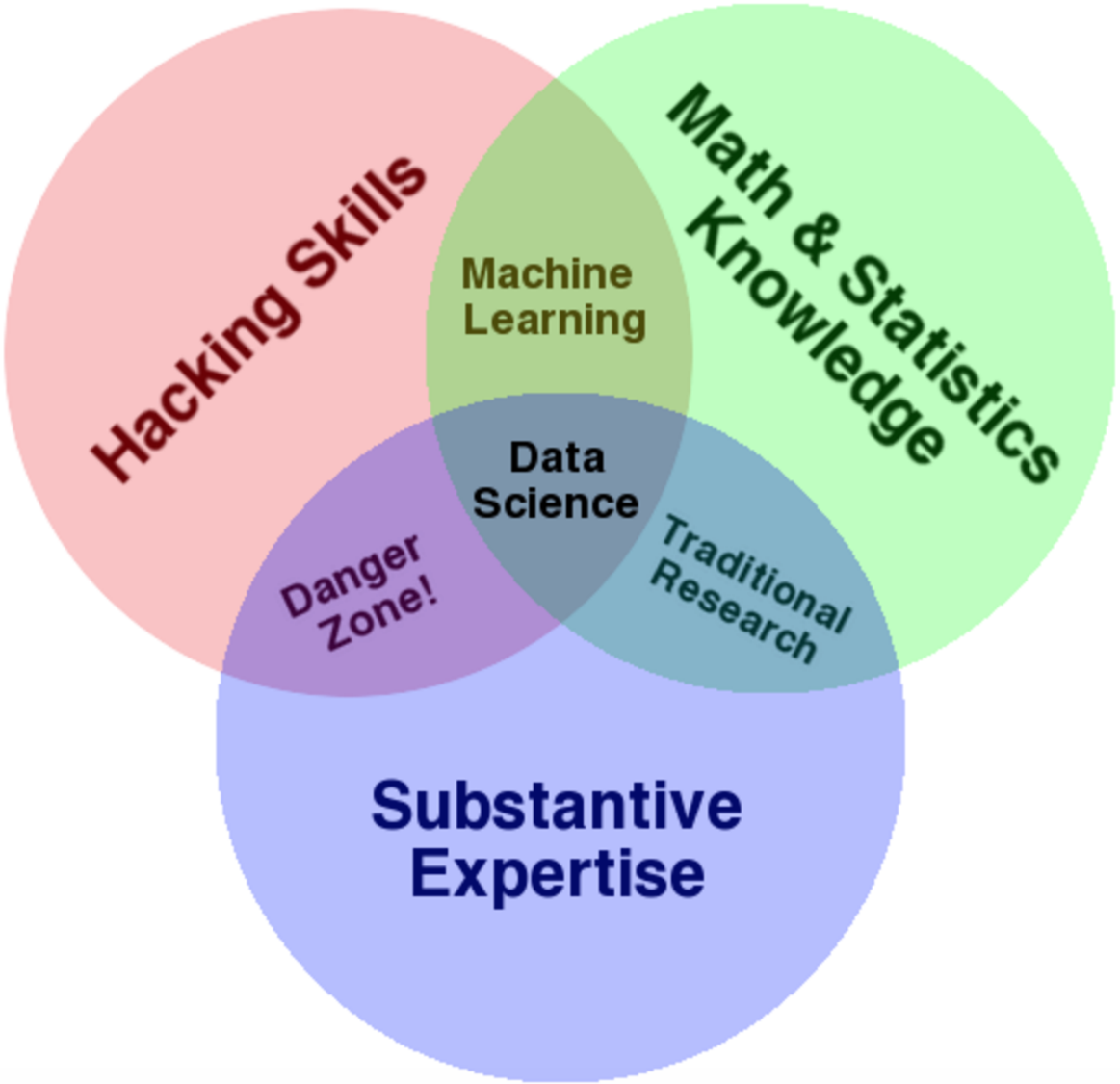
Overview

1. Introduction
2. Approximation Theory
3. Sigmoid Networks
4. Rectification Networks

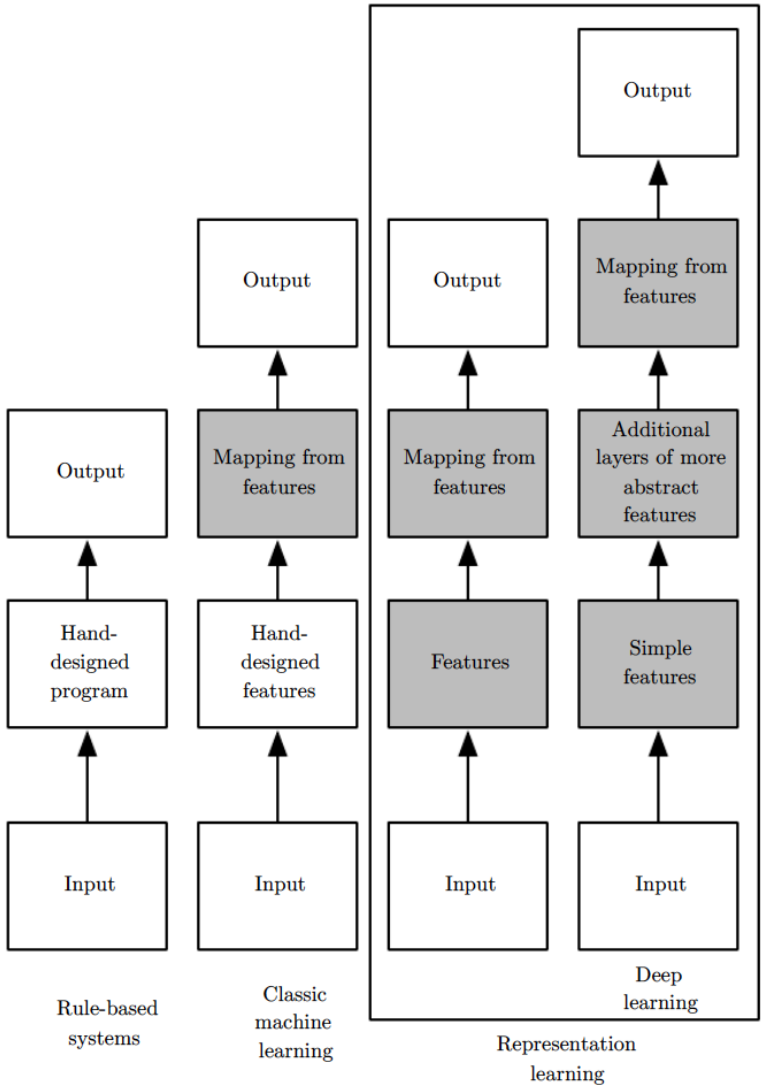
Section 1

Introduction

Conway's Data Science Venn Diagram

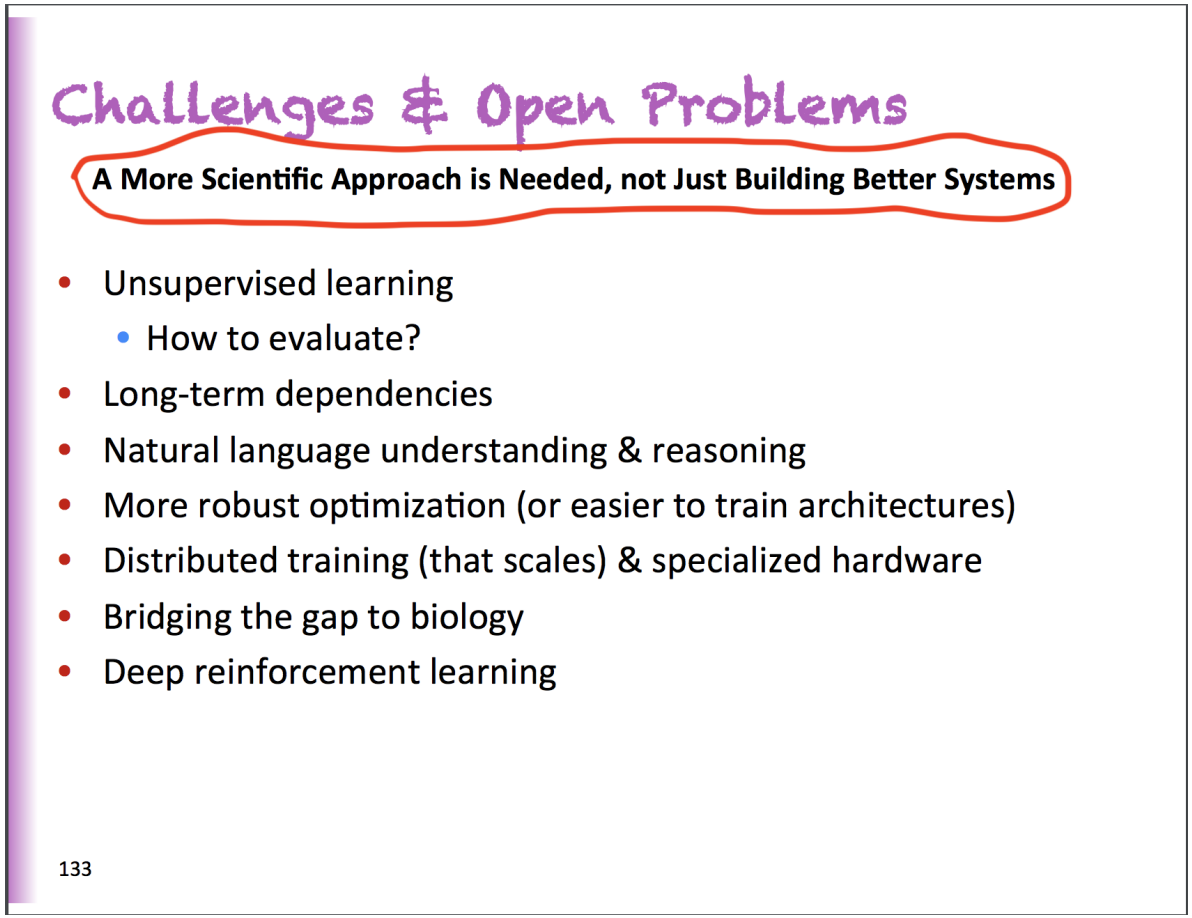


From Rules to (Deep) Machine Learning



(From DL Fig 1.5)

Last Slide from Hinton, Bengio and Lecun's Deep Learning Tutorial at NIPS 2015



Challenges & Open Problems

A More Scientific Approach is Needed, not Just Building Better Systems

- Unsupervised learning
  - How to evaluate?
- Long-term dependencies
- Natural language understanding & reasoning
- More robust optimization (or easier to train architectures)
- Distributed training (that scales) & specialized hardware
- Bridging the gap to biology
- Deep reinforcement learning

133

Roadmap for today's lecture

- Can Deep Neural Networks Learn Anything?
- If so, Why They Do Not Overfit? Or Do They?

Section 2

Approximation Theory

Ridge Functions

Definition: $f : \mathbb{R}^n \to \mathbb{R}$ is a ridge function, if it can be written as

$$f(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b),$$

for some choice of $\sigma : \mathbb{R} \to \mathbb{R}$, $\mathbf{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$.

- denote the linear part of f by $\bar{f}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$, then

$$L_f(c) = \bigcup_{d \in \sigma^{-1}(c)} L_{\bar{f}}(d)$$

- if σ is differentiable at $z = \mathbf{w}^\top \mathbf{x} + b$, then

$$\nabla_{\mathbf{x}} f \overset{\text{chain rule}}{=} \sigma'(z) \nabla_{\mathbf{x}} \bar{f} \overset{\text{directly}}{=} \sigma'(z)\, \mathbf{w}$$

Theorem: Let $f : \mathbb{R}^n \to \mathbb{R}$ differentiable at $\mathbf{x}$. Then either $\nabla f(\mathbf{x}) = 0$ or $\nabla f(\mathbf{x}) \perp L_f(f(\mathbf{x}))$.

Dense Approximations

Definition: Dense Subsets

A function class $\mathcal{H} \subseteq C(\mathbb{R}^d)$ is dense in $C(\mathbb{R}^d)$, if and only if

$$\forall f \in C(\mathbb{R}^d) \;\; \forall \epsilon > 0 \;\; \forall K \subset \mathbb{R}^d, \text{ compact:}$$

$$\exists h \in \mathcal{H} \;\; \text{s.t.} \;\; \max_{\mathbf{x} \in K} |f(\mathbf{x}) - h(\mathbf{x})| = \|f - h\|_{\infty, K} < \epsilon$$

- uniform approximation on compacta (i.e. use of ∞-norm)
- $\sup \to \max$ (Bolzano-Weierstrass)
- informally speaking: we can approximate any continuous f to arbitrary accuracy (on K) with a suitable member of $\mathcal{H}$

Universal Approximation with Ridge Functions

Definitions: Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a scalar function

$$\mathcal{G}_\sigma^n := \{g : g(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b) \text{ for some } \mathbf{w} \in \mathbb{R}^n,\, b \in \mathbb{R}\}$$

$$\mathcal{G}^n := \bigcup_{\sigma \in C(\mathbb{R})} \mathcal{G}_\sigma^n, \quad \text{universe of continuous ridge functions}$$

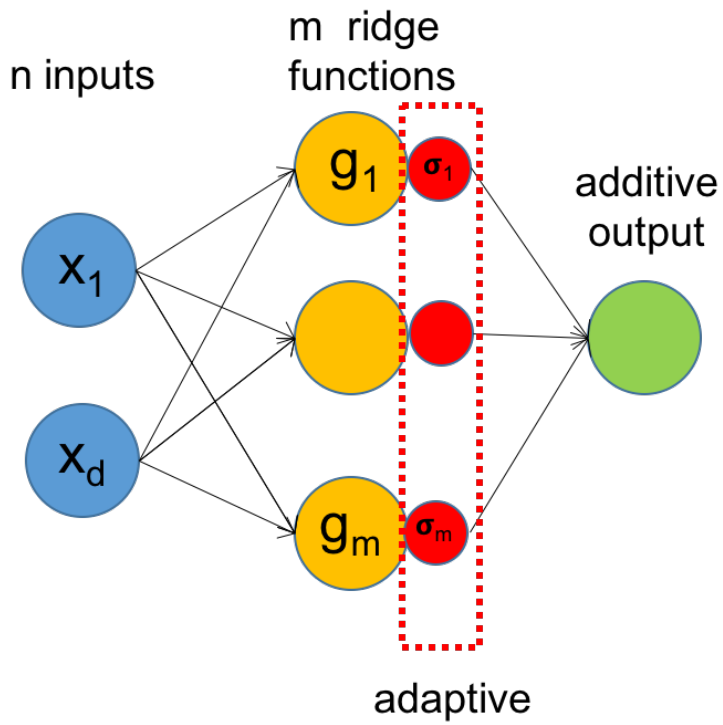
Theorem: Vostrecov and Kreines, 1961

$\mathcal{H}^n := \text{span}(\mathcal{G}^n)$ is dense in $C(\mathbb{R}^n)$.

- note that $\mathcal{H}^n = \{h : h = \sum_{j=1}^r g_j,\ g_j \in \mathcal{G}^n\}$
  (can absorb linear combination weights in functions g_j)

Ridge Function Network

Theorem uses additive combinations of arbitrary (unspecified) ridge functions.



- would require some adaptivity of the non-linearity (= learning the activation function)
- not inconceivable, but not commonly done

Special Ridge Function Families

Can we get away with special ridge function families?

More precisely, can we chose one $\sigma \in C(\mathbb{R})$ such that $\mathcal{G}_\sigma^n$ is dense in $C(\mathbb{R}^n)$.

We also investigate how to simplify the discussion to the case of $n = 1$.

Approximation Theorem

Theorem: Lechno, Lin, Pinkus, and Schocken 1993

Let $\sigma \in C^\infty(\mathbb{R})$, not a polynomial, then $\mathcal{H}_\sigma^1$ is dense in $C(\mathbb{R})$.

Corollary: MLPs with one hidden layer and any non-polynomial, smooth activation function are universal function approximators.

Lemma: MLPs with one hidden layer and a polynomial activation function are **not** universal function approximators.

Remark: Smoothness requirement can be substantially weakened. See previous results on rectified activation functions.

Approximation Theorem: Proof

1. For all $h \neq 0$,

$$\frac{\sigma((\lambda + h)t + \theta) - \sigma(\lambda t + \theta)}{h} \in \mathcal{H}_\sigma$$

2. It follows that (generalizing to all k-th derivatives)

$$\frac{d^k}{d\lambda^k}\sigma(\lambda t + \theta)\Big|_{\lambda=0} = t^k \sigma^{(k)}(\theta) \in \mathsf{cl}(\mathcal{H}_\sigma)$$

3. If we can show that there always is a θ_0 such that $\sigma^{(k)}(\theta_0) \neq 0$ then we are be guaranteed that $t^k \in \mathsf{cl}(\mathcal{H}_\sigma)$ and hence all polynomials.

4. By the Weierstrass theorem this implies the result. □

Approximation Theorem: Proof

The remaining gap follows by the following result.

Theorem (after Donoghue, 1969)

If σ is C^∞ on $(a;b)$ and it is not a polynomial thereon, then there exists a point $\theta_0 \in (a;b)$ such that $\sigma^{(k)}(\theta_0) \neq 0$ for $k = 0, 1, 2, \ldots$.

Lemma: Dimension Lifting

Hard to work in $C(\mathbb{R}^n)$. Can we work with $C(\mathbb{R})$ instead? Yes!

Lemma: Pinkus 1999 (simplified)

The density of $\mathcal{H}_\sigma^1$ in $C(\mathbb{R})$ with

$$\mathcal{H}_\sigma^1 := \mathsf{span}(\mathcal{G}_\sigma^1) = \mathsf{span}\{\sigma(\lambda t + \theta) : \lambda, \theta \in \mathbb{R}\}$$

implies the density of

$$\mathcal{H}_\sigma^n := \mathsf{span}(\mathcal{G}_\sigma^n) = \mathsf{span}\{\sigma(\mathbf{w}^\top \mathbf{x} + b) : \mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R}\}\,.$$

in $C(\mathbb{R}^n)$ for any $n \geq 1$.

- informally: we can lift density property of ridge function families from $C(\mathbb{R})$ to $C(\mathbb{R}^n)$.

Lemma: Proof Sketch To modify

1. Given a compact set $K \in \mathbb{R}^n$, $f \in C(K)$, Since $\mathcal{H}^n$ is dense in $C(\mathbb{R})^n$ then, there exist some r and for any $\epsilon > 0$:

$$\left| f(\mathbf{x}) - \sum_{i=1}^{r} g_i(\mathbf{x}) \right| < \frac{\epsilon}{2} \qquad \forall \mathbf{x} \in K$$

   with $g_i(\mathbf{x}) = \sigma_i(\mathbf{w}_i^\top \mathbf{x} + b_i)$.

2. By compactness of K one has $\{\mathbf{w}_i^\top \mathbf{x} + b_i : \mathbf{x} \in K\} \subseteq [\alpha_i, \beta_i]$

3. For each i: $g_i(t)$ for $t \in [\alpha_i, \beta_i]$, which is a compact set, hence:

$$\left| g_i(t) - \sum_{i=1}^{m_i} c_{ij} \sigma(\lambda_{ij} t + \theta_{ij}) \right| < \frac{\epsilon}{2r} \qquad \forall \mathbf{x} \in K$$

Lemma: Proof Sketch cont'

4. Putting it together yields:

$$\left| f(\mathbf{x}) - \sum_{i=1}^{r} \sum_{i=1}^{m_i} c_{ij} \sigma(\lambda_{ij}(\mathbf{w}_i^\top \mathbf{x} + b_i) + \theta_{ij}) \right| < \epsilon \qquad \forall \mathbf{x} \in K$$

□

Ridge Functions: Key Advantage

Key advantage of ridge functions

Picking out a direction of change: done in linear part
$\Longrightarrow$ essentially equivalent to linear case

Non-linear activation function: models rate of change in the chosen direction. Just a $C(\mathbb{R})$ function, dimension independent.

Continuous activation function can be approximated by expansions with fixed activation function (examples follow). Simplification at the cost of increased layer width.

Summary

Answers

- Continuous functions can be well-approximated by linear combinations of ridge functions (universal function approximation)
- Justifies use of computational units which apply a scalar non-linearity to a linear function of the inputs

Questions

- What do we know about the "size" of the representation? *Unfortunately, very little*
- How do achieve these approximation and how do we measure their quality?

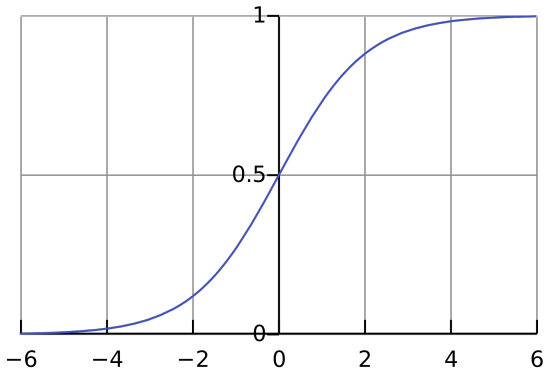
Section 3

Sigmoid Networks

Sigmoid Functions

What are "good" activation functions?

Sigmoid activation function:
logistic function (or $\tanh$)



$$\sigma(t) = \frac{1}{1+e^{-t}} = \frac{e^t}{1+e^t} \in (0;1), \quad \sigma^{-1}(\mu) = \ln\left(\frac{\mu}{1-\mu}\right)$$

$$\tanh(t) = 2\sigma(2t) - 1 \in (-1;1)$$

Sigmoidal MLP: Approximation Guarantees

We will now state and discuss without proof (a simplified) version of the famous result of (Barron, 1993)[1], which relates the residual to the number of sigmoidal neurons in the (single) hidden layer.

Consider a multi-layer perceptron, where:

$$\tilde{f}_m(\mathbf{x}) = \sum_{j=1}^{m} \alpha_j \, \sigma\left(\mathbf{w}_j^\top \mathbf{x} + b_j\right) + \beta$$

with σ is a bounded (measurable) and monotonic function such that $\sigma(t) \overset{t\to\infty}{\longrightarrow} 1$ and $\sigma(t) \overset{t\to-\infty}{\longrightarrow} 0$. (Barron's version of "sigmoid")

[1] Universal approximation bounds for superpositions of a sigmoidal functions

Sigmoidal MLP: Approximation Guarantees

Theorem: Barron, 1993, simplified

For every $f : \mathbb{R}^n \to \mathbb{R}$ w/ absolutely continuous Fourier transform and for every m there is a function of the form $\tilde{f}_m$ such that

$$\int_{B_r} (f(\mathbf{x}) - \tilde{f}_m(\mathbf{x}))^2 \mu(d\mathbf{x}) \leq O(1/m)$$

where $B_r = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| \leq r\}$ and μ is any probability measure on B_r.

Comments:

- most remarkably, the residual bound does not depend on the input dimension n
- proof uses iterative process of adding units

Section 4

Rectification Networks

Rectified Linear Units

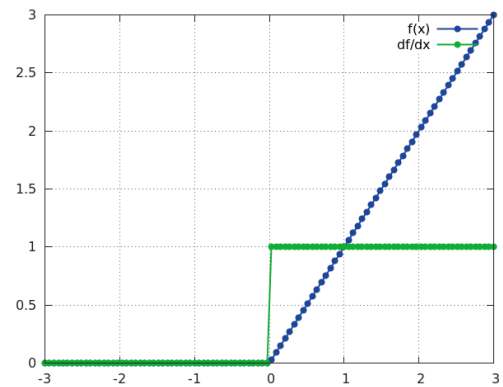
"Modern school", rectified activation function

Definition: ReLU (rectified linear unit)

The activation function of a ReLU is defined as

$$(x)_+ := \max(0, x), \qquad \underbrace{\partial (x)_+}_{\text{subdifferential}} = \begin{cases} \{1\} & \text{if } x > 0 \\ \{0\} & \text{if } x < 0 \\ [0; 1] & \text{if } x = 0 \end{cases}$$

- linear function over half-space $\mathcal{H}$
- zero on complement $\mathcal{H}^c = \mathbb{R}^n - \mathcal{H}$
- non-smooth, but simple subdifferential



Rectified Linear Units

Closely related alternative activation function

Definition: Absolute value rectification

The activation function of an absolute value unit (AVU) is given by

$$|x| := \begin{cases} x, & \text{if } x \geq 0 \\ -x & \text{otherwise} \end{cases}, \quad \partial|x| = \begin{cases} 1 & \text{if } x > 0 \\ [-1;1] & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

- relation to ReLU activation

$$(x)_+ = \frac{x + |x|}{2}, \quad \text{and} \quad |x| = 2(x)_+ - x$$

Best Practice: Rectification in Computer Vision

What is the Best Multi-Stage Architecture for Object Recognition? (Jarret, Kavukcuoglu, Ranzato, LeCun 2009)

- *"The surprising answer is that using a rectifying non-linearity is the single most important factor in improving the performance of a recognition system."*
- experimental results

Two Stage System: $[64.F^{9\times9}_{CSG} - R/N/P^{5\times5}] - [256.F^{9\times9}_{CSG} - R/N/P^{4\times4}]$ - log_reg					
	$R_{abs} - N - P_A$	$R_{abs} - P_A$	$N - P_M$	$N - P_A$	P_A
U^+U^+	65.5%	60.5%	61.0%	34.0%	32.0%
R^+R^+	64.7%	59.5%	60.0%	31.0%	29.7%
UU	63.7%	46.7%	56.0%	23.1%	9.1%
RR	62.9%	33.7%($\pm$1.5)	37.6%($\pm$1.9)	19.6%	8.8%
GT	55.8%				

- uses $|x|$, but similar results for $(x)_+$

Rectification in the 19th Century

Background: proof of Weierstrass theorem (see Pinkus, 2000)

Any $f \in C[0;1]$ can be uniformly approximated to arbitrary precision by a polygonal line (cf. Shektman, 1982)

Lebesgue (1898): polygonal line with m pieces can be written

$$g(x) = \underbrace{ax + b}_{\text{linear}} + \sum_{i=1}^{m-1} c_i (x - x_i)_+$$

- knots: $0 = x_0 < x_1 < \cdots < x_{m-1} < x_m = 1$
- $m + 1$ parameters $a, b, c_i \in \mathbb{R}$
- ReLU function approximation in 1d

Rectification in the 19th Century

Proof (sketch, by induction over m):

$m = 1$: Linear function over $[0; 1] \Longrightarrow$ fit line with a, b

$m \Longrightarrow m + 1$: Given $m + 1$ knots and values (x_j, y_j). Eliminate knot x_m and chose a, b and c_i $(i < m)$ to fit this function (induction hypothesis).

Now modify c_{m-1} and chose c_m to fit a wedge to the three points (x_{m-1}, y_{m-1}), (x_m, y_m), and $(x_{m+1} = 1, y_{m+1})$
(trivial, left as an excercise).

Rectification in the 19th Century

Alternative representation with absolute value function

$$g(x) = a'x + b' + \sum_{i=1}^{m-1} c'_i |x - x_i|$$

Comments:

- Weierstrass: $C[0; 1]$ functions can be uniformly approximated by polynomials
- Lebesgue: proof for Weierstrass theorem by showing that $|x|$ can be uniformly approximated on $[-1; 1]$ by polynomials
- in a way, piecewise linear functions may be the "most natural" way to think about function approximation

Approximation by Rectified Networks

Theorem

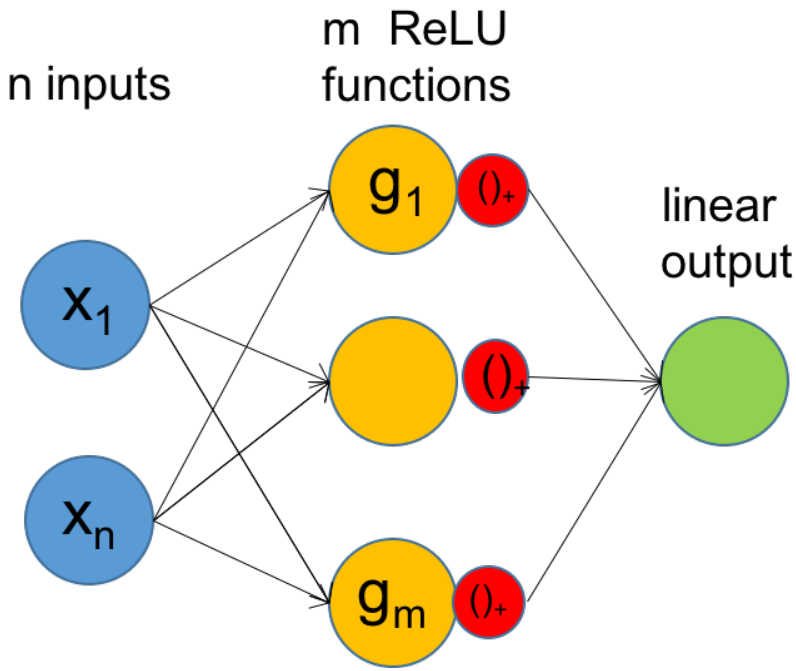
Networks with one hidden layer of ReLU or absolute value units are universal function approximators.

Proof sketch

1. Universally approximate $C(K)$ functions (K, compact) by polygonal lines
2. Represent polygonal lines by (linear function $+$) linear combinations of $(\cdot)_+$- or $|\cdot|$-functions
3. Apply dimension lifting lemma to show density of the linear span of resulting ridge function families $\mathcal{G}^n_{(\cdot)_+}$ and $\mathcal{G}^n_{|\cdot|}$

ReLU Network

Theorem allows for the use of restricted set of ridge functions (e.g. ReLU).



- no adaptivity of the non-linearity required (=fixed)
- possibly at the price of increasing hidden layer width (m)

Piecewise Linear Functions

ReLU and AVU define a piecewise linear function with 2 pieces.

$\mathbb{R}^n$ is partitioned into two open half spaces (and a border face):

$$H^+ := \{\mathbf{x} : \mathbf{w}^\top \mathbf{x} + b > 0\} \subset \mathbb{R}^n$$
$$H^- := \{\mathbf{x} : \mathbf{w}^\top \mathbf{x} + b < 0\} \subset \mathbb{R}^n$$
$$H^0 := \mathbb{R}^n - H^+ - H^-$$

Note that

$$g_{(\cdot)_+}\left(H^0\right) = g_{|\cdot|}\left(H^0\right) = 0$$

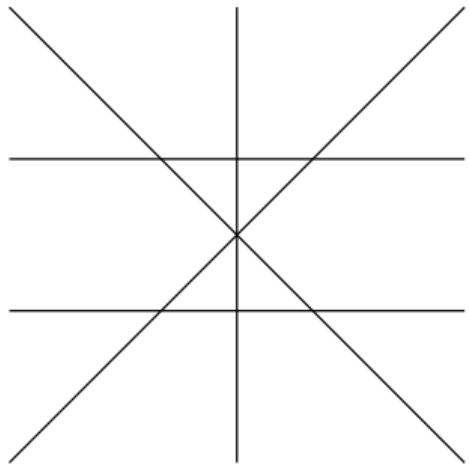
$$g_{(\cdot)_+}\left(H^-\right) = 0$$

$$g_{|\cdot|}(\mathbf{x}) = g_{|\cdot|}(\mathbf{v} - \mathbf{x}), \quad \text{with} \quad \mathbf{v} = -2b\frac{\mathbf{w}}{\|\mathbf{w}\|^2} \ \text{(mirroring at } \mathbf{w})$$

Linear Combinations of Rectified Units

Question: by linearly combining m rectified units, into how many $(= R(m))$ cells is $\mathbb{R}^n$ maximally partitioned?

Example: 5 lines 14 cells



(cf. R.P. Stanley, An Introduction to Hyperplane Arrangements, 2006)

Linear Combinations of Rectified Units

Question: by linearly combining m rectified units, into how many $(= R(m))$ cells is $\mathbb{R}^n$ maximally partitioned?

Explicit formula, e.g. by Zaslavsky 1975:

$$R(m) \leq \sum_{i=0}^{\min\{m,n\}} \binom{m}{i}$$

- note that for $m \leq n$, $R(m) = 2^m$ (exponential growth)
- for given n, asymptotically, $R(m) \in \Omega(m^n)$
  - i.e. there is a polynomial slow-down, which is induced by the limitation of the input space dimension

Deep Combinations of Rectified Units

Question: Process n inputs through L ReLU layers with widths $m_1, \ldots, m_L \in O(m)$. Into how many $(= R(m, L))$ cells can $\mathbb{R}^n$ be maximally partitioned?

Theorem: Montufar et al, 2014

With the definitions as above it holds asymptotically that

$$R(m, L) \in \Omega\left(\left(\frac{m}{n}\right)^{n(L-1)} m^n\right)$$

- essentially: for any fixed n, exponential growth can be ensured by making layers sufficiently wide $(m > n)$ and increasing the level of functional nesting (i.e. depth L).

Deep Combinations of Rectified Units

Examples: Montufar et al, 2014

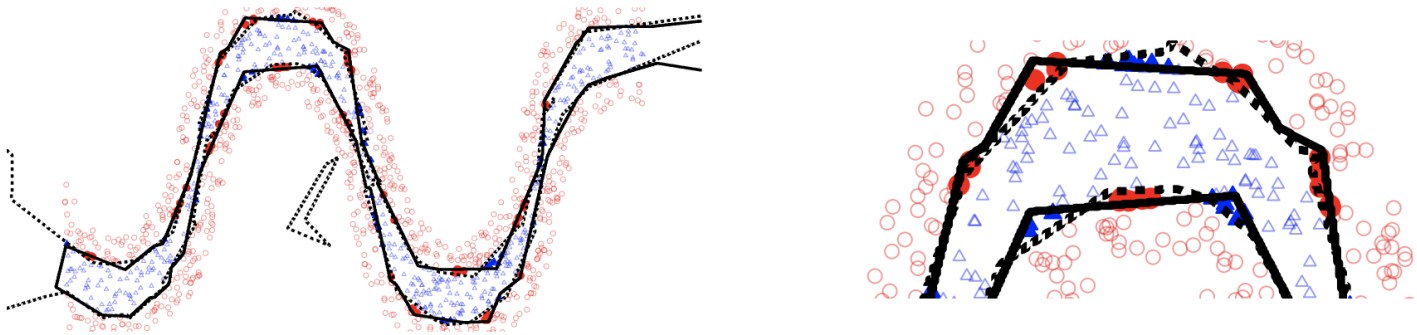


Figure 1: Binary classification using a shallow model with 20 hidden units (solid line) and a deep model with two layers of 10 units each (dashed line). The right panel shows a close-up of the left panel. Filled markers indicate errors made by the shallow model.

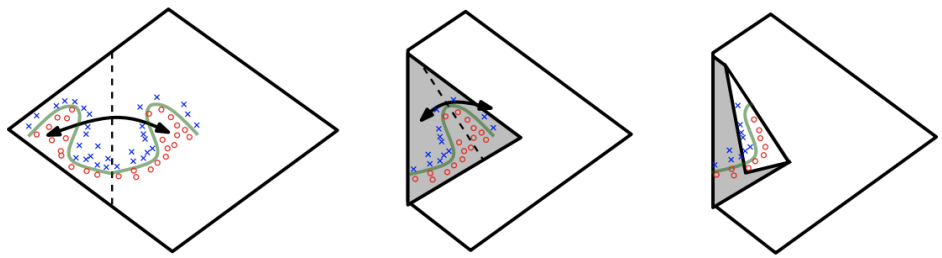


Figure 3: Space folding of 2-D space in a non-trivial way. Note how the folding can potentially identify symmetries in the boundary that it needs to learn.

Hinging Hyperplanes

Another piecewise linear set of functions (cf. Breiman, 1993)

Definition: Hinge function

If $g : \mathbb{R}^n \to \mathbb{R}$ can be written with parameters $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^n$ and $b_1, b_2 \in \mathbb{R}$ as below it is called a hinge function

$$g(\mathbf{x}) = \max\left(\mathbf{w}_1^\top \mathbf{x} + b_1, \mathbf{w}_2^\top \mathbf{x} + b_2\right)$$

- two hyperplanes, "glued" together at the face $(\mathbf{w}_1 - \mathbf{w}_2)^\top \mathbf{x} + (b_1 - b_2) = 0$
- easy to fit single hinging hyperplane (iterative algorithm)
- representational power: $2\max(f, g) = f + g + |f - g|$.

Hinging Hyperplanes (Wang & Sun, 2005)

- Given a Continuous Piecewise linear (PWL) function in $\mathbb{R}^n$, we can represent the function as:

$$
\begin{aligned}
n = 1 \quad & \mathbf{w}_1^\top \mathbf{x} + b_1 \pm \left| \mathbf{w}_2^\top \mathbf{x} + b_2 \right| \\
n = 2 \quad & \mathbf{w}_1^\top \mathbf{x} + b_1 \pm \left| \mathbf{w}_2^\top \mathbf{x} + b_2 \right| \pm \left| \mathbf{w}_3^\top \mathbf{x} + b_3 + \left| \mathbf{w}_4^\top \mathbf{x} + b_4 \right| \right| \\
n = 3 \quad & \mathbf{w}_1^\top \mathbf{x} + b_1 \pm \left| \mathbf{w}_2^\top \mathbf{x} + b_2 \right| \pm \left| \mathbf{w}_3^\top \mathbf{x} + b_3 + \left| \mathbf{w}_4^\top \mathbf{x} + b_4 \right| \right| \\
& \pm \left| \mathbf{w}_5^\top \mathbf{x} + b_5 + \left| \mathbf{w}_6^\top \mathbf{x} + b_6 + \left| \mathbf{w}_7^\top \mathbf{x} + b_7 \right| \right| \right| \\
\dots \quad & \dots
\end{aligned}
$$

- for a continuous PWL function in $\mathbb{R}^n$ we need n nested absolute value functions.

Hinging Hyperplanes (Wang & Sun, 2005)

Definition: k-Hinge Functions

$$g(\mathbf{x}) = \max\left(\mathbf{w}_1^\top \mathbf{x} + b_1, \ldots, \mathbf{w}_k^\top \mathbf{x} + b_k\right)$$

Theorem: Wang and Sun, 2005

Every continuous piecewise linear function from $\mathbb{R}^n \to \mathbb{R}$ can be written as a signed sum of k-Hinges with $k_i \leq \lceil \log_2(n+1) \rceil$.

$$\sum_i \theta_i g_i(\mathbf{x}) \qquad \theta_i \in \{\pm 1\}$$

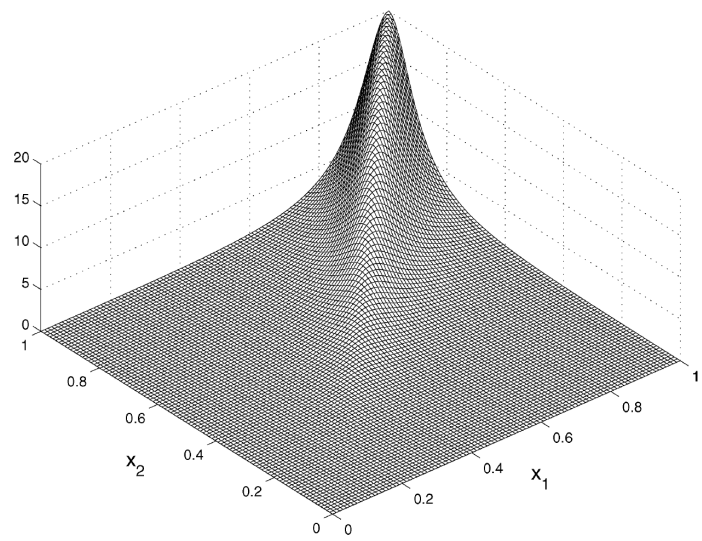
- Reducing the growth of absolute value nesting to logarithmic growth, instead of linear.
- PWL functions are dense and can approximate any function to any desired degree.

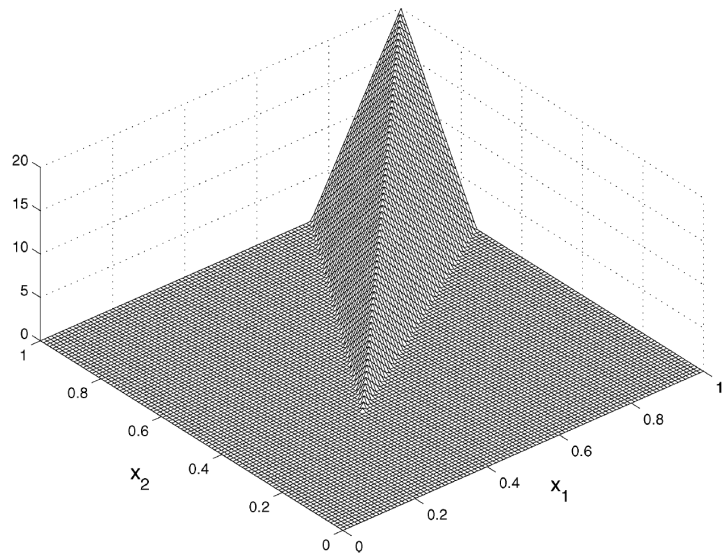
Hinging Hyperplanes (Wang & Sun, 2005)

Though a generalized canonical representation can represent all CPWL functions, it is unpractical to use it in black-box modeling, for it is very difficult to identify a desired high-level nested absolute-value function from sampled data. In addition, the aforementioned papers provide only an upper bound on the number of nestings necessary for representing all CPWL functions. It is not known whether or not the same goal can be attained using less nestings of absolute-value functions.

The main contribution of this correspondence is to provide a general representation whose structure is much simpler than that of a generalized canonical representation. Hence it is likely to be used in black-box modeling. This general representation is constructed simply by adding a sufficient number of linear functions to current hinges. Specifically

Hinging Hyperplanes: Example





$$f(\mathbf{x}) = \frac{0.2e^{-4((x_1-1)^2+(x_2-1)^2)}}{(x_1-x_2)^2+0.01} \qquad p(\mathbf{x}) = \begin{cases} 80x_1 - 50x_2 - 10, & \mathbf{x} \in \Omega_1 \\ 80x_2 - 50x_1 - 10, & \mathbf{x} \in \Omega_2 \\ 0, & \mathbf{x} \in \Omega_2 \end{cases}$$

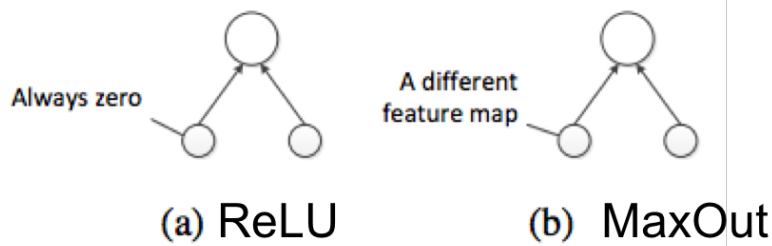
$$p(\mathbf{x}) = \max\{65(x_1 - x_2), 65(x_2 - x_1), 15(x_1 + x_2) - 10\} - \max\{65(x_1 - x_2), 65(x_2 - x_1)\}$$

Maxout Networks

- re-discovery of k-Hinges: Maxout (Goodfellow et al, 2013)
- Maxout: max (non-linearity) applied to groups of (linear) functions
- Partition $[1:d]$ into sets A_j, then define

$$G_j(\mathbf{x}) := \max_{i \in A_j} \{\mathbf{w}_i^\top \mathbf{x} + b_i\}$$

- Literature
  - cf. DL, pp. 188
  - Goodfellow et al, Maxout Networks, ICML 2013



(from Cai et al. 2013)

2 x Maxout = Allout :)

Theorem: Goodfellow, 2013

Maxout networks with two maxout units are universal function approximators.

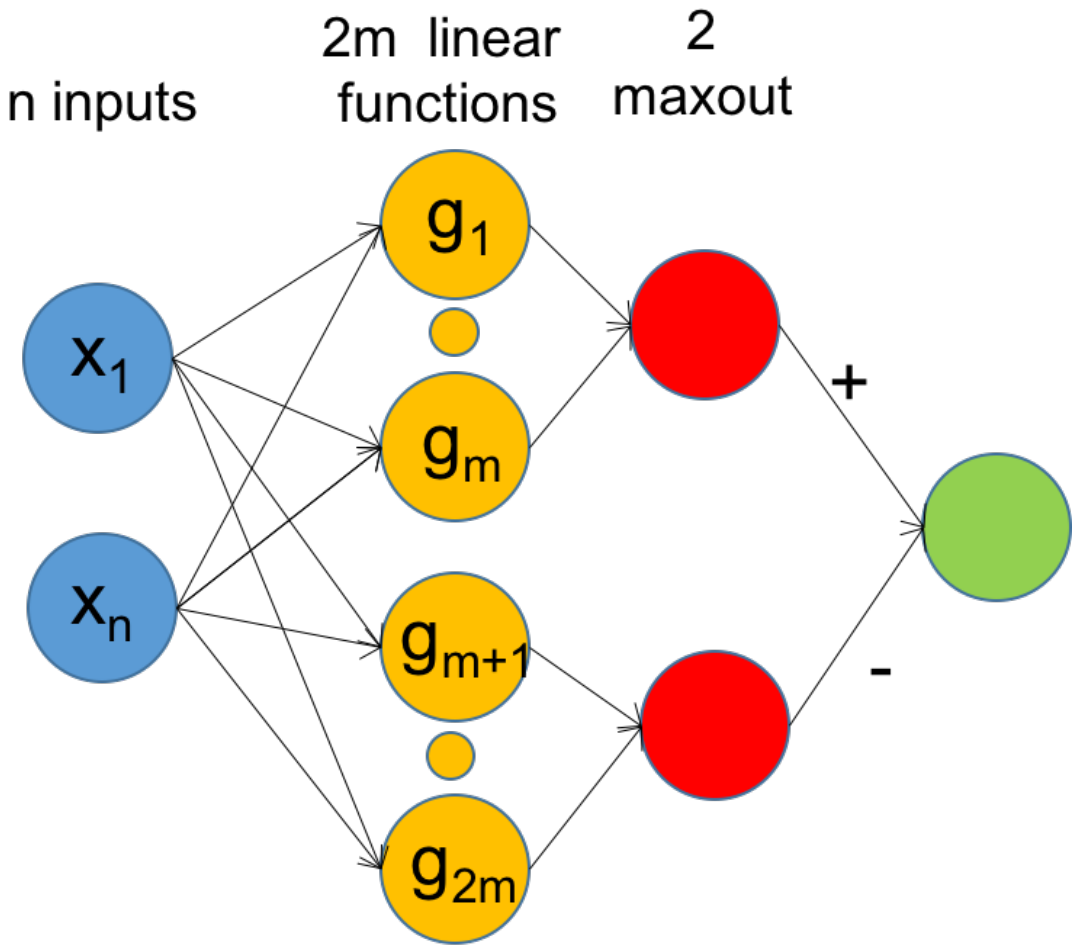
Proof sketch:

1. Wang's theorem: linear network with two maxout units and a linear output unit (subtraction) can represent any continuous PWL function (exactly!).

2. Continuous PWL functions are dense in $C(\mathbb{R}^n)$.

Comment: most minimalistic use of non-linearity

Minimalistic 2 x Maxout Network



In practice: more than 2 maxout units.

Best Practice: Maxout in Speech Recognition

Cai et al, Deep maxout neural networks for speech recognition, ASRU Workshop 2013

Table 3. Speech recognition results using 300 hours of Switchboard training data. All networks have 7 hidden layers with 2048 units each. Performances are measured in WER given in %.

Model	Hub5'00-SWB	RT03S-FSH
sigmoid	15.7	19.0
ReLU	15.3	18.7
maxout	**15.1**	**18.5**

- higher improvements on smaller data sets
- faster training

Bibliography

- A. Pinkus, (1999). "Approximation Theory of the MLP model in Neural Networks". Acta Numerica pp 143-195.
- K. Jarrett, K. Kavukcuoglu, M. A. Ranzato and Y. LeCun, (2009). "What is the Best Multi-Stage Architecture for Object Recognition?" ICCV.
- G. Montufar, R. Pascanu, K. Cho and Y. Bengio, (2014). "On the Number of Linear Regions of Deep Neural Networks". NIPS.
- S. Wang and X. Sun, (2005). "Generalization of Hinging Hyperplanes". IEEE IT 51(12).