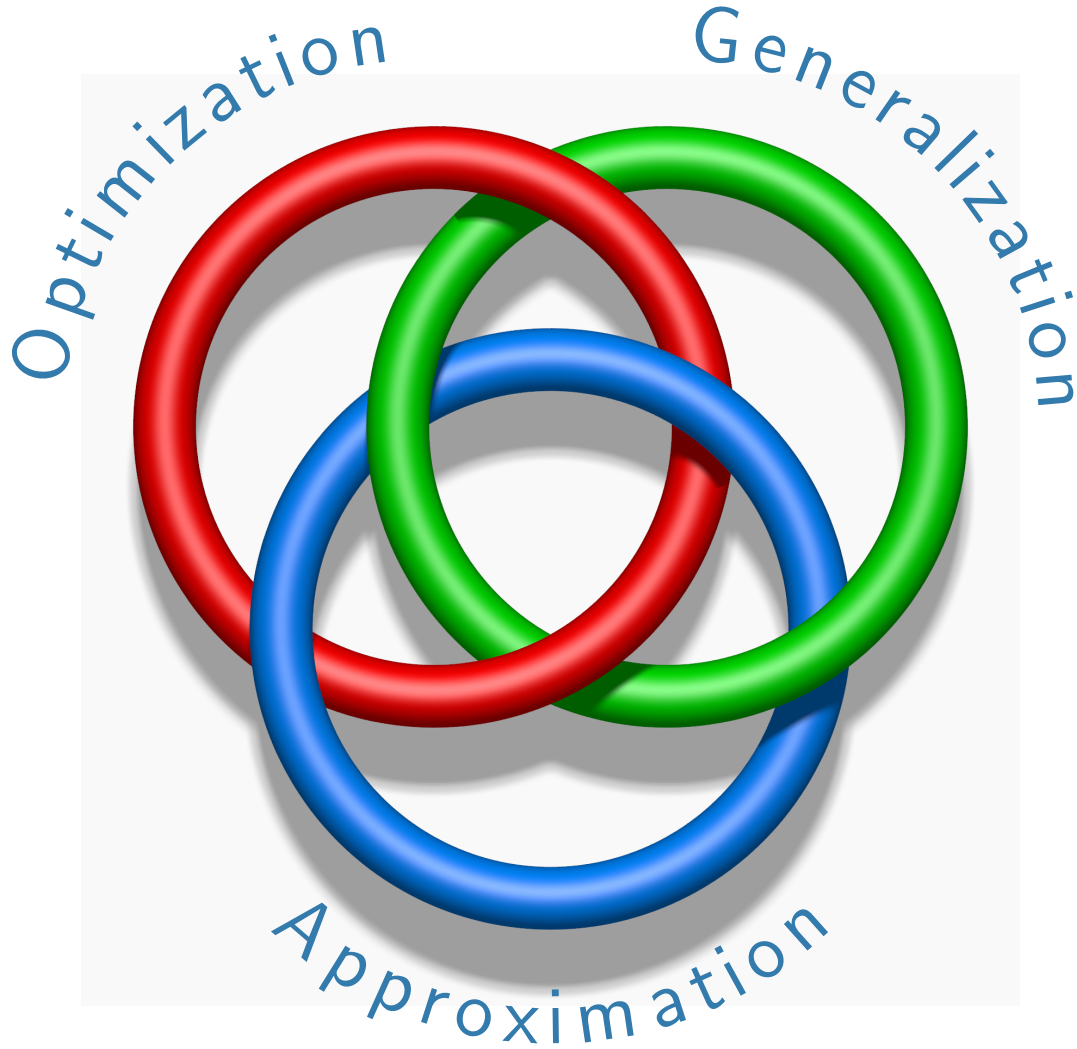
Neural Networks Math

Supervised Learning and Approximation Properties

Guido Montúfar
montufar@math.ucla.edu

Transylvanian Machine Learning Summer School, July 2018

UCLA


Optimization
Generalization
Approximation

Lots of exciting math I won't cover here!

- Generative models, reinforcement learning
- Connections of parametrization, optimization landscapes, optimization techniques, and generalization

1. Observations
   - The curse of dimensionality
   - Multivariate non-linear functions
   - Is learning feasible?

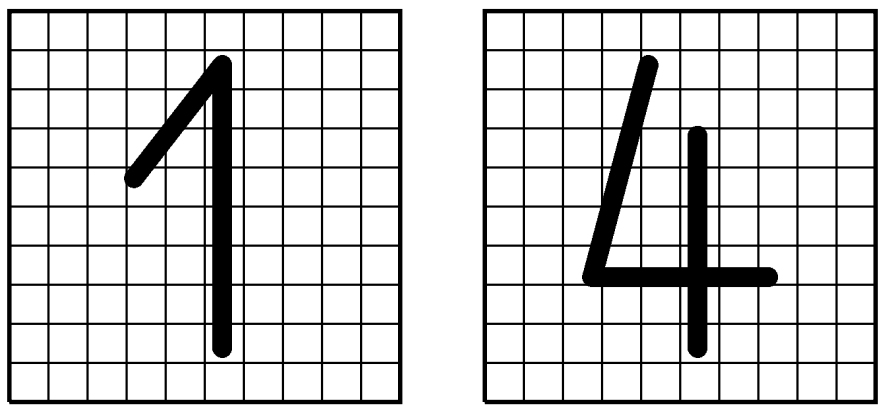
2. Supervised Learning Theory
   - A formal definition of learning
   - Growth function and VC dimension
   - Linear threshold networks
   - Geometric techniques
   - General bounds for neural networks

3. Approximation Properties
   - Universal approximation and rate of convergence
   - Deep and Shallow
   - Combinatorial views on ReLUs

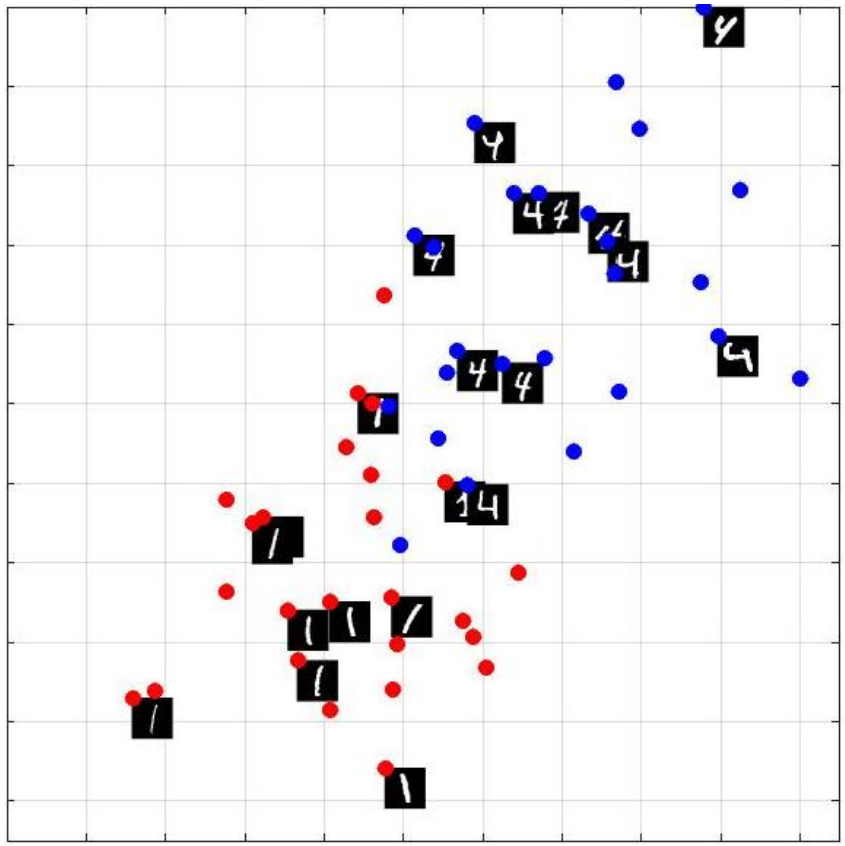
The curse of dimensionality

We follow Chapter 1 in [Bis95].

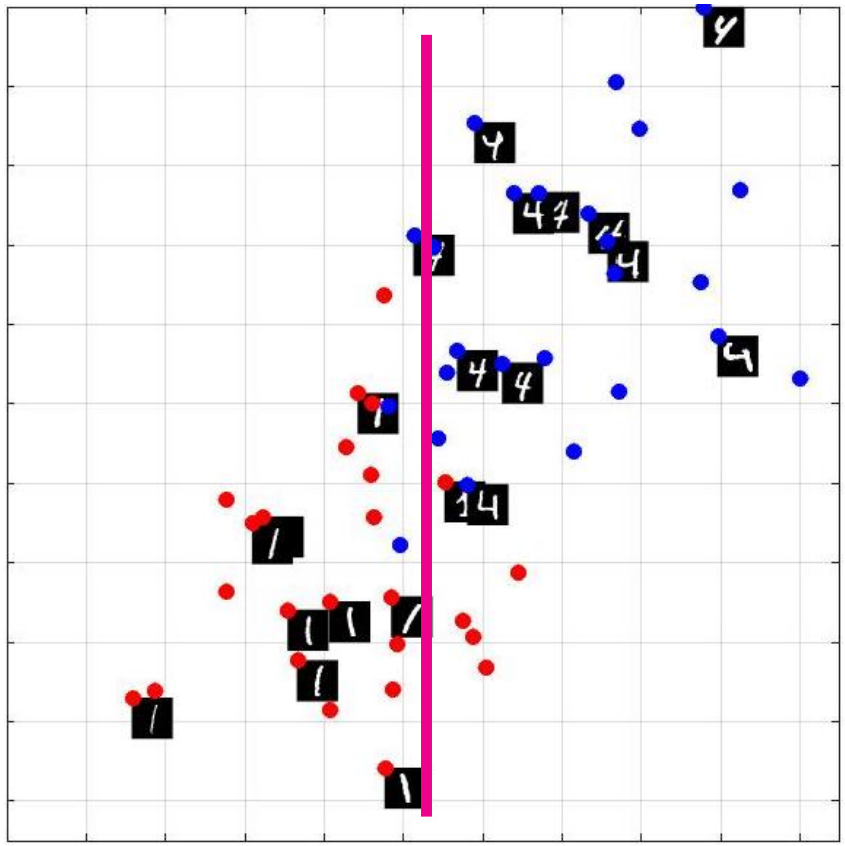
- Consider a simple problem of distinguishing handwritten versions of the digits ‘1’ and ‘4’.



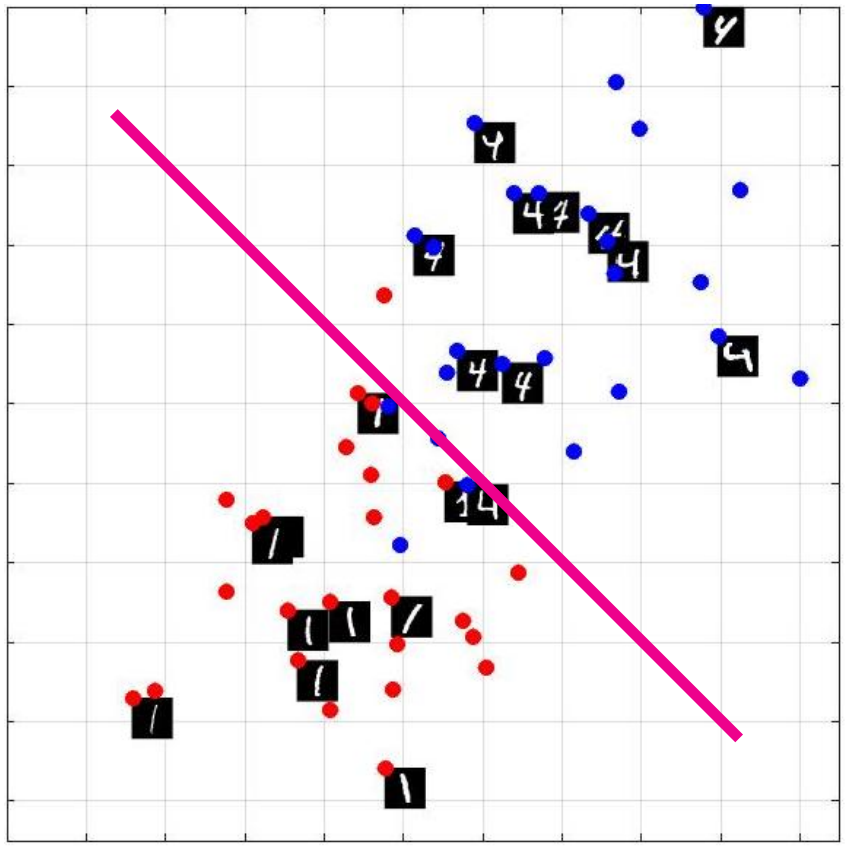
- Having two features instead of just one, can improve the performance of a classification system.



- Having two features instead of just one, can improve the performance of a classification system.

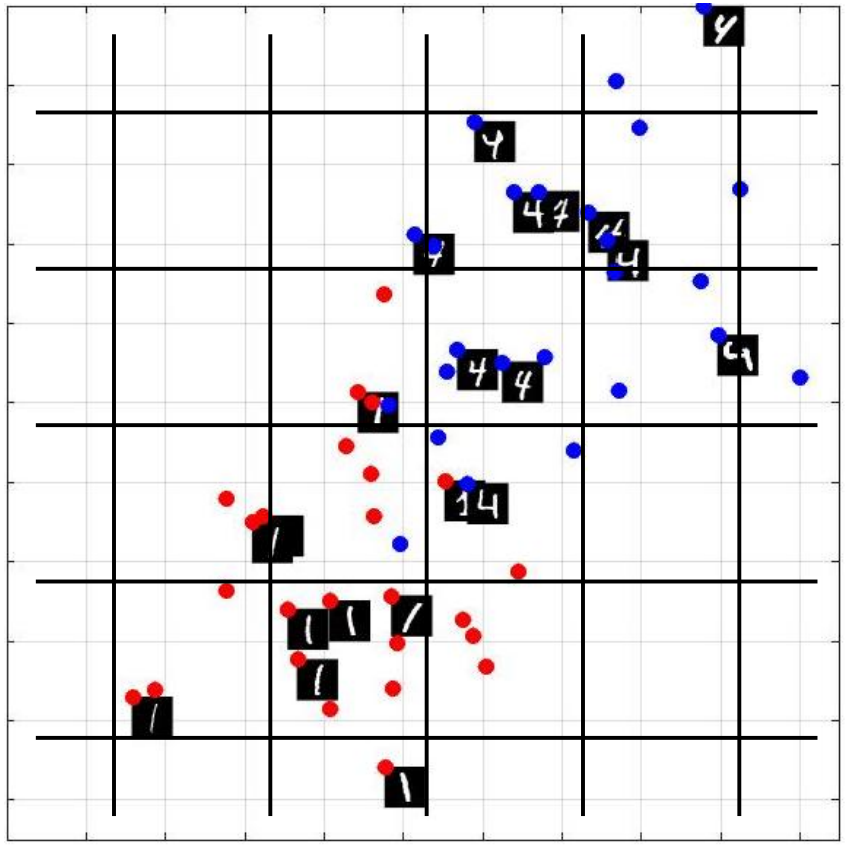


- Having two features instead of just one, can improve the performance of a classification system.



- So why not use more and more features? In practice we find that, beyond a certain point, this can worsen the performance.
- In order to understand this, consider following technique for modeling mappings on the basis of training data.

- Divide the range of each input variable into a number of intervals. For a new point x, we return the average value y for all training points which lie in the same box of input space.



- By increasing the number K of intervals for each variable, we can obtain a more precise specification. But K^d cells is exponential in the dimensionality of the input space.
- Specifying the mapping requires an exponential number of examples! This phenomenon is called curse of dimensionality.
- If we only have/can process a limited amount of data, increasing the dimensionality of the space rapidly leads to very sparse data, and the above gives a very poor representation.

- Neural networks can be much less susceptible to the curse of dimensionality, exploiting important properties of real data:

1. The input variables are generally correlated in some way, so that the data points do not fill out the entire input space, but tend to be restricted to a sub-space of lower dimension.
2. For most mappings of practical interest, the output varies smoothly with the input. Thus it is possible to infer the output values at intermediate points where no data is available, by a process similar to interpolation.

Multivariate non-linear functions

- The role of neural networks is to provide general parametrized non-linear mappings between inputs and outputs.
- Polynomials provide such mappings too. Provided we have a sufficiently large number of terms in the polynomial, we can approximate any reasonable function to arbitrary accuracy. This suggests that we could use polynomials

$$y = w_0 + \sum_{\lambda \subseteq [n]} w_\lambda \prod_{i \in \lambda} x_i$$

- For an Nth order polynomial, the number of parameters would grow as d^N. This is polynomial in d, rather than exponential, but it is still very rapid growth.

- The importance of neural networks lies in the way in which they deal with the problem of scaling with dimensionality.
- The hidden units are adapted to the data, and so their number only needs to grow as the complexity of the problem grows, and not simply as the dimensionality of the input.
- The number of free parameters in such models, for a given number of hidden functions, typically grows linearly, or quadratically, with the dimensionality of the input space (compared with the d^N for general degree N polynomials).

- For neural networks, classic works show that the residual sum of squares error falls as $O(1/N)$, where N is the number of hidden units, irrespective of the number if input variables. (This uses dimension dependent smoothness assumptions)
- For polynomials the error only decreases as $O(1/N^{2/d})$, where d is the dimension of the input space. Similar results hold for other series expansions/linear combinations of fixed functions.
- The price payed for this efficient scaling is that the network functions are non linear in the adaptive parameters. Unlike polynomial fitting, determining the parameters now presents a number of complications, such as local minima. (Optimization landscape of neural networks is a hot topic.)

Is learning feasible?

We follow [AMMIL12].

- Say we managed to classify all data points in our training set correctly. What about new data points?

Training data

$$\begin{bmatrix}101\\110\\001\end{bmatrix} \quad \begin{bmatrix}101\\011\\100\end{bmatrix} \quad \begin{bmatrix}110\\010\\001\end{bmatrix} \quad +1$$

$$\begin{bmatrix}010\\101\\010\end{bmatrix} \quad \begin{bmatrix}000\\111\\000\end{bmatrix} \quad \begin{bmatrix}000\\010\\000\end{bmatrix} \quad -1$$

We follow [AMMIL12].

- Say we managed to classify all data points in our training set correctly. What about new data points?

Training data

$$\begin{bmatrix}101\\110\\001\end{bmatrix} \quad \begin{bmatrix}101\\011\\100\end{bmatrix} \quad \begin{bmatrix}110\\010\\001\end{bmatrix} \quad +1$$

$$\begin{bmatrix}010\\101\\010\end{bmatrix} \quad \begin{bmatrix}000\\111\\000\end{bmatrix} \quad \begin{bmatrix}000\\010\\000\end{bmatrix} \quad -1$$

Test data

$$\begin{bmatrix}100\\010\\001\end{bmatrix} \quad ?$$

We follow [AMMIL12].

- Say we managed to classify all data points in our training set correctly. What about new data points?

Training data

$$\begin{bmatrix}101\\110\\001\end{bmatrix} \quad \begin{bmatrix}101\\011\\100\end{bmatrix} \quad \begin{bmatrix}110\\010\\001\end{bmatrix} \quad +1$$

$$\begin{bmatrix}010\\101\\010\end{bmatrix} \quad \begin{bmatrix}000\\111\\000\end{bmatrix} \quad \begin{bmatrix}000\\010\\000\end{bmatrix} \quad -1$$

Test data

$$\begin{bmatrix}100\\010\\001\end{bmatrix} \quad ?$$

- Learning is only possible in a probabilistic sense

1. Observations
   - The curse of dimensionality
   - Multivariate non-linear functions
   - Is learning feasible?

2. Supervised Learning Theory
   - A formal definition of learning
   - Growth function and VC dimension
   - Linear threshold networks
   - Geometric techniques
   - General bounds for neural networks

3. Approximation Properties
   - Universal approximation and rate of convergence
   - Deep and Shallow
   - Combinatorial views on ReLUs

A formal definition of learning

We follow Chapter 2 in [AB09].

- A training sample of length m is an element

$$z = (z_1, \dots, z_m) = ((x_1, y_1), \dots, (x_m, y_m)) \in \mathcal{Z}^m = (\mathcal{X} \times \mathcal{Y})^m.$$

  We assume that this is distributed according to some unknown product distribution P^m.

- Given a function $h\colon \mathcal{X} \to \mathcal{Y}$, the error of h with respect to P is

$$\mathrm{er}_P(h) = P\{(x, y) \in \mathcal{Z}\colon h(x) \neq y\}.$$

- The sample error of h on a sample $z \in \mathcal{Z}^m$ is

$$\hat{\mathrm{er}}_z(h) = \frac{1}{m}|\{i \in \{1, \dots, m\}\colon h(x_i) \neq y_i\}|.$$

(More generally we consider the expectation value of some loss function $l(h(x), y)$.)

Given some data z generated from P and a class of functions $\mathcal{H}\colon \mathcal{X} \to \mathcal{Y}$ our aim is to produce some $g \in \mathcal{H}$ with

$$\mathrm{er}_P(g) < \mathrm{opt}_P(\mathcal{H}) + \epsilon,$$

where

$$\mathrm{opt}_P(\mathcal{H}) = \inf_{h \in \mathcal{H}} \mathrm{er}_P(h).$$

This can be formalized as follows. We focus on the case $\mathcal{Y} = \{0, 1\}$.

Definition 1

Suppose that $\mathcal{H}$ is a set of functions $\mathcal{X} \to \{0,1\}$. A *learning algorithm* L for $\mathcal{H}$ is a function

$$L\colon \cup_{m=1}^{\infty} \mathcal{Z}^m \to \mathcal{H}$$

so that, given any $\epsilon \in (0,1)$ and $\delta \in (0,1)$, there is an $m_0(\epsilon,\delta)$ with

$$P^m\left(\mathrm{er}_P(L(z)) < \mathrm{opt}_P(\mathcal{H}) + \epsilon\right) \geq 1-\delta$$

for any $m \geq m_0(\epsilon,\delta)$ for any P on $\mathcal{Z} = \mathcal{X} \times \{0,1\}$.

- The sample complexity of a learning algorithm L is

$$m_L(\epsilon, \delta) = \min\{m\colon\ m \text{ sufficient sample size for } (\epsilon, \delta)\text{-learning } \mathcal{H} \text{ with } L\}$$

- The inherent sample complexity for learning with $\mathcal{H}$ is

$$m_{\mathcal{H}}(\epsilon, \delta) = \min_L m_L(\epsilon, \delta),$$

where the min is over the learning algorithms L for $\mathcal{H}$.

- Our aim is to produce a function h in $\mathcal{H}$ that has near minimal error $\mathrm{er}_P(h)$.
- Given that the true errors of the functions are unknown, it is natural to use the sample errors as estimates.
- We can consider a Sample Error Minimization (SEM) algorithm that for given data z chooses a function $L(z)$ with

$$\hat{\mathrm{er}}_z(L(z)) = \min_{h \in \mathcal{H}} \hat{\mathrm{er}}_z(h)$$

Hoeffding's inequality gives the following.

Theorem 2

Suppose h is a function $\mathcal{X} \to \{0,1\}$. Then

$$P^m \left\{ |\hat{\mathrm{er}}_z(h) - \mathrm{er}_P(h)| \geq \epsilon \right\} \leq 2\exp(-2\epsilon^2 m),$$

for any probability distribution P, any ϵ, and any positive integer m.

A simple union bound gives following uniform convergence result.

Theorem 3

Suppose that $\mathcal{H}$ is a finite set of functions $\mathcal{X} \to \{0,1\}$. Then

$$P^m \left\{ \max_{h \in \mathcal{H}} |\hat{\text{er}}_z(h) - \text{er}_P(h)| \geq \epsilon \right\} \leq 2|\mathcal{H}| \exp(-2\epsilon^2 m),$$

for any probability distribution P, any ϵ, and any positive integer m.

Uniform error estimation

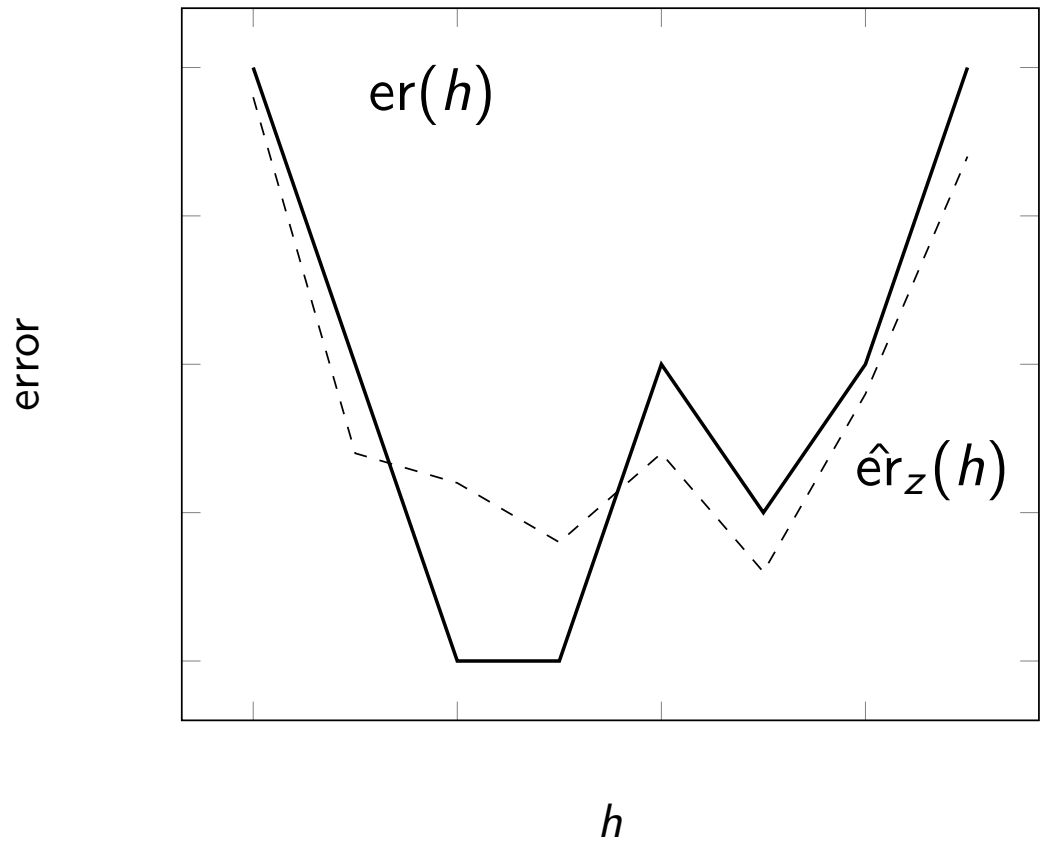


Figure 1: Empirical and real error across hypotheses.

Since the sample error tracks the error uniformly over $\mathcal{H}$, we can learn by minimizing the sample error.

Theorem 4 (SEM learning finite models)

Suppose that $\mathcal{H}$ is a finite set of functions $\mathcal{X} \to \{0,1\}$. Let $L\colon \cup_{m=1}^{\infty} \mathcal{Z}^m \to \mathcal{H}$ be such that for any m and any $z \in \mathcal{Z}^m$,

$$\hat{\mathrm{er}}_z(L(z)) = \min_{h\in\mathcal{H}} \hat{\mathrm{er}}_z(h).$$

Then L is a learning algorithm for $\mathcal{H}$, with estimation error

$$\epsilon_L(m,\delta) \leq \left(\frac{2}{m}\ln\frac{2|\mathcal{H}|}{\delta}\right)^{1/2},$$

and sample complexity

$$m_L(\epsilon,\delta) \leq \frac{2}{\epsilon^2}\ln\frac{2|\mathcal{H}|}{\delta}.$$

- We gave a formal definition of learning and discussed how learning is feasible for finite hypothesis sets $\mathcal{H}$.
- However, most interesting function classes are not finite. As it turns out, learning is also feasible for many function classes of this type, provided they are not too complex.
- Next we examine two complexity measures, the growth function and the VC-dimension, and how they determine the inherent sample complexity of the learning problem.

Growth function and VC dimension

We follow Chapter 3 in [AB09].

- Consider a finite subset $\mathcal{S} \subseteq \mathcal{X}$ of the input space. For a function class $\mathcal{H}$ the restriction to the set $\mathcal{S}$ is denoted $\mathcal{H}|_{\mathcal{S}}$.
- We can view the cardinality of $\mathcal{H}|_{\mathcal{S}}$ (and how it compares to $2^{|\mathcal{S}|}$) as a measure of the classification complexity of $\mathcal{H}$ with respect to the set $\mathcal{S}$.

Definition 5 (Growth function)

The *growth function* of $\mathcal{H}$ is the function $\Pi_{\mathcal{H}} : \mathbb{N} \to \mathbb{N}$ defined by

$$\Pi_{\mathcal{H}}(m) = \max\left\{|\mathcal{H}|_{\mathcal{S}}| : \mathcal{S} \subseteq \mathcal{X}, |\mathcal{S}| = m\right\}.$$

Note that $\Pi_{\mathcal{H}}(m) \leq 2^m$ for all m. If $\mathcal{H}$ is finite, then clearly $\Pi_{\mathcal{H}}(m) \leq |\mathcal{H}|$ for all m, with equality for sufficiently large m.

A *dichotomy* is one of the functions $f : \mathcal{S} \to \{0, 1\}$ in $\mathcal{H}|_{\mathcal{S}}$.

Definition 6 (Simple perceptron)

A *simple perceptron* is a model of functions $\mathcal{X} = \mathbb{R}^n \to \{0,1\}$ of the form

$$f : x \mapsto \sigma(w^\top x - \theta),$$

parametrized by weights $w \in \mathbb{R}^n$ and bias $\theta \in \mathbb{R}$. Here

$$\sigma : s \mapsto \begin{cases} 1 & \text{if } s \geq 0 \\ 0 & \text{otherwise .} \end{cases}$$

Theorem 7 (Growth function of the simple perceptron)

Let $\mathcal{H}$ be the set of functions computable by a simple perceptron with $n \in \mathbb{N}$ real inputs. Then

$$\Pi_{\mathcal{H}}(m) = 2\sum_{k=0}^{n} \binom{m-1}{k}.$$

We can divide the parameter space into regions on which the same dichotomy is computed on the subset $\mathcal{S}$ of the input space $\mathcal{X}$.

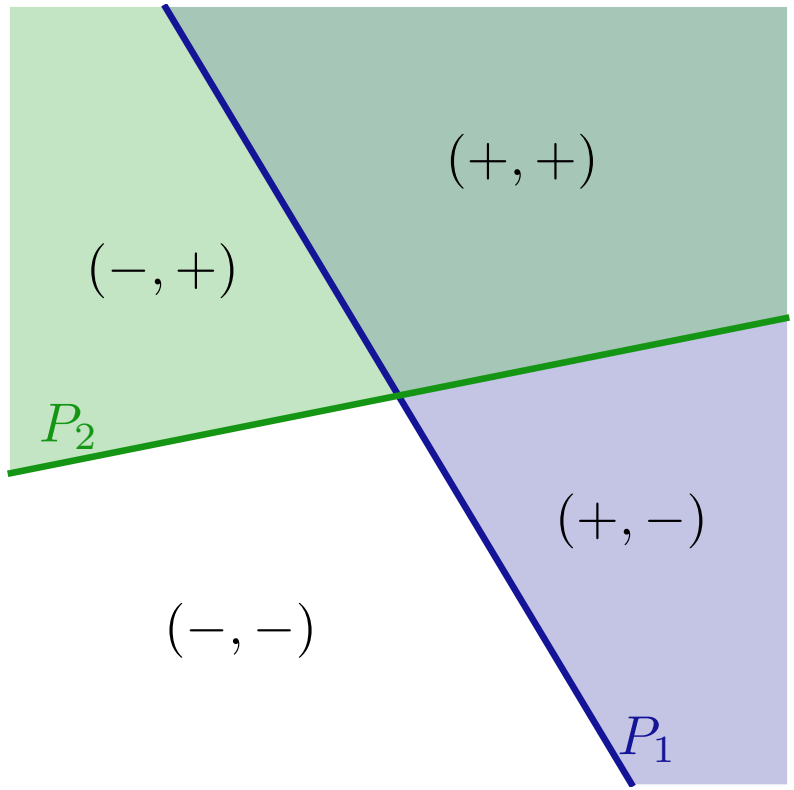


Figure 2: The hyperplane $P_i = \{(w, \theta) \in \mathbb{R}^2 \colon w^\top x_i - \theta = 0\}$ consists of parameters for which x_i lies at the decision boundary of the perceptron.

Lemma 8 (Simple perceptron and central arrangement)

For a set $\mathcal{S} = \{x_1, \dots, x_m\} \subseteq \mathbb{R}^n$, let $P_1, \dots, P_m$ be the hyperplanes defined by

$$P_i = \{(w, \theta) \in \mathbb{R}^{n+1} \colon w^\top x_i - \theta = 0\}.$$

Then

$$|\mathcal{H}|_{\mathcal{S}}| = r(\mathbb{R}^{n+1} \setminus \bigcup_{i=1}^{m} P_i).$$

Here $r(U)$ denotes the number of connected components of U.

Lemma 9 (Regions of a general position central arrangement)

Consider any $m, d \in \mathbb{N}$. *Let* $T = \{\tilde{x}_1, \dots, \tilde{x}_m\} \subseteq \mathbb{R}^d$ *be such that every subset* d *of them are linearly independent. Let* $P_i = \{v \in \mathbb{R}^d : v^\top \tilde{x}_i = 0\}$, $i = 1, \dots, m$. *Then*

$$r(\mathbb{R}^d \setminus \bigcup_{i=1}^{m} P_i) = 2 \sum_{k=0}^{d-1} \binom{m-1}{k}.$$

Proof of Lemma 9. We proceed by induction.

- Fix d and consider an arrangement of m hyperplanes, $P_1, \dots, P_m$.
- If we add one more hyperplane P_{m+1}, we add as many regions to the total, as the number of regions in P_{m+1} generated by $P_{m+1} \cap P_i$, $i = 1, \dots, m$.
- This means that $r(m+1, d) = r(m, d) + r(m, d-1)$.
- The hypothesis holds for $m = 1$ and any d, since $r(1, d) = 2 = 2 \sum_{k=0}^{d-1} \binom{0}{k}$.
  It also holds for $d = 1$ and any m, since $r(m, 1) = 2 = 2 \sum_{k=0}^{0} \binom{m-1}{k}$.

- Assuming that the hypothesis holds for (m, d) and $(m, d-1)$, we now prove that it holds for $(m+1, d)$, which implies that it holds for all (m, d). We have

$$
\begin{aligned}
r(m+1,d) &= r(m,d) + r(m,d-1) \\
&= 2\sum_{k=0}^{d-1}\binom{m-1}{k} + 2\sum_{k=0}^{d-2}\binom{m-1}{k} \\
&= 2\sum_{k=1}^{d-1}\left(\binom{m-1}{k-1} + \binom{m-1}{k}\right) + \binom{m-1}{0} \\
&= 2\sum_{k=1}^{d-1}\binom{m}{k} + 1 = 2\sum_{k=0}^{d-1}\binom{m}{k}.
\end{aligned}
$$

Definition 10 (Shattering)

Consider a function class $\mathcal{H}$ and a set $\mathcal{S}$ of m points in input space $\mathcal{X}$. If $\mathcal{H}$ can compute all dichotomies of $\mathcal{S}$ (i.e., if $|\mathcal{H}|_{\mathcal{S}}| = 2^m$), we say that $\mathcal{H}$ *shatters* $\mathcal{S}$.

Definition 11 (VC-dimension)

The *Vapnik-Chervonenkis dimension* (VC-dimension) of $\mathcal{H}$ is the size of the largest shattered subset of $\mathcal{X}$,

$$\text{VCdim}(\mathcal{H}) := \max\{m\colon \Pi_{\mathcal{H}}(m) = 2^m\}.$$

As we shall see, the behavior of the growth function is tightly constrained by the VC-dimension.

Now, the VC-dimension constraints the growth function.

Theorem 12

For a function class $\mathcal{H}$ with $\mathrm{VCdim}(\mathcal{H}) = d$,

$$\Pi_{\mathcal{H}}(m) \leq \sum_{i=0}^{d} \binom{m}{i}.$$

The previous theorem implies that for a function class $\mathcal{H}$ with $\text{VCdim}(\mathcal{H}) = d$ the growth function is at most polynomial in m.

Corollary 13

For a function class $\mathcal{H}$ with $\text{VCdim}(\mathcal{H}) = d$, if $m \geq d$, then

$$d \leq \log_2 \Pi_{\mathcal{H}}(m) \leq d \log_2(em/d).$$

This is because

$$\sum_{i=0}^{d} \binom{m}{i} = \begin{cases} 2^m, & m \leq d \\ < \left(\frac{em}{d}\right)^d, & m > d \end{cases}$$

Learning finite VC dimension

We follow Chapters 4 and 5 in [AB09].

Theorem 14

Let $\mathcal{H}$ be a set of functions $\mathcal{X} \to \{0,1\}$ with $\mathrm{VCdim}(\mathcal{H}) = d \geq 1$. Let L be a sample error minimizing algorithm for $\mathcal{H}$, so that $\hat{\mathrm{er}}_z(L(z)) = \min_{h\in\mathcal{H}} \hat{\mathrm{er}}_z(h)$. Then L is a learning algorithm for $\mathcal{H}$. If $m \geq d/2$, then

$$\epsilon_L(m,\delta) \leq \epsilon_0(m,\delta) = \left(\frac{32}{m}\left(d \ln \frac{2em}{d} + \ln \frac{4}{\delta}\right)\right)^{1/2}$$

and

$$m_L(\epsilon,\delta) \leq m_0(\epsilon,\delta) = \frac{64}{\epsilon^2}\left(2d \ln \frac{12}{\epsilon} + \ln \frac{4}{\delta}\right).$$

Uniform convergence

- Using SEM is motivated by the idea that sample errors track the true errors.
- The crucial step to obtain the learnability result is to show uniform convergence of sample errors to true errors.

Theorem 15

Let $\mathcal{H}$ be a set of functions $\mathcal{X} \to \{0,1\}$. Let P be a distribution on $\mathcal{Z} = \mathcal{X} \times \{0,1\}$. For $\epsilon \in (0,1)$ and $m \in \mathbb{N}$ we have

$$P^m \left\{ |\operatorname{er}_P(h) - \hat{\operatorname{er}}_z(h)| \geq \epsilon \text{ for some } h \in \mathcal{H} \right\} \leq 4\Pi_{\mathcal{H}}(2m) \exp\left(-\frac{\epsilon^2 m}{8}\right).$$

- This is one of the central results in learning theory.

- The proof involves 3 steps: Symmetrization, Permutations, Reduction.
- We need to bound the probability that a sample z of length m is 'bad', meaning that $|\hat{\text{er}}_z(h) - \text{er}_P(h)| \geq \epsilon$ for some $h \in \mathcal{H}$.
- To this end, we consider samples $z = rs$ and compare $\hat{\text{er}}_r(h)$ and $\hat{\text{er}}_z(h)$.

Lower bound for learning

Lemma 16

Let $0 < \epsilon < 1$. *Let* α *be uniformly distributed on* $\{\alpha_-, \alpha_+\}$, *where* $\alpha_- = \frac{1-\epsilon}{2}$ *and* $\alpha_+ = \frac{1+\epsilon}{2}$. *Let* $\xi_1, \ldots, \xi_m$ *be iid binary variables with* $\Pr(\xi = 1) = \alpha$. *Let* f *be a function from* $\{0,1\}$ *to* $\{\alpha_-, \alpha_+\}$. *Then*

$$\Pr(f(\xi_1, \ldots, \xi_m) \neq \alpha) > \frac{1}{4}\left(1 - \sqrt{1 - \exp(\frac{-2\lceil m/2 \rceil \epsilon^2}{1-\epsilon^2})}\right).$$

In particular, if this probability is at most δ, *where* $0 < \delta < 1/4$, *then*

$$m \geq 2\left\lfloor \frac{1-\epsilon^2}{2\epsilon^2} \ln(\frac{1}{8\delta(1-2\delta)}) \right\rfloor.$$

- Here we think of α as some hidden binary variable, and ξ_i as a probabilistic binary observation which equals α with probability $\frac{1+\epsilon}{2}$.
- We think of f as a hypothesis for α, given the observations $\xi_1, \ldots, \xi_m$.
- The lemma discusses the probability that the hypothesis disagrees with α. As the lemma shows, the probability of disagreement is bounded below, depending on the 'distance' $\frac{\epsilon}{2}$ between observations, and the amount of data m.

Theorem 17 (Complexity of learning)

Let $\mathcal{H}$ be a function class from $\mathcal{X}$ to $\{0,1\}$, containing at least two functions, and having finite VC-dimension d. For any learning algorithm L for $\mathcal{H}$, the sample complexity of L satisfies

$$m_L(\epsilon,\delta) \geq \frac{1}{320}\frac{d}{\epsilon^2}$$

for all $0 < \epsilon,\ \delta < 1/64$. Furthermore,

$$m_L(\epsilon,\delta) \geq 2\left\lfloor \frac{1-\epsilon^2}{2\epsilon^2}\ln\left(\frac{1}{8\delta(1-2\delta)}\right)\right\rfloor$$

for all $0 < \epsilon < 1$ and $0 < \delta < 1/4$.

VC dimension quantifies sample complexity

Theorem 18

Let $\mathcal{H}$ be a set of functions $\mathcal{X} \to \{0,1\}$. Then $\mathcal{H}$ is learnable if and only if $\mathrm{VCdim}(\mathcal{H}) = d < \infty$. *There are constants $c_1, c_2 > 0$ with*

$$\frac{c_1}{\epsilon^2}\left(d + \ln\frac{1}{\delta}\right) \leq m_{\mathcal{H}}(\epsilon,\delta) \leq \frac{c_2}{\epsilon^2}\left(d + \ln\frac{1}{\delta}\right)$$

for all $\epsilon \in (0, 1/40)$ and $\delta \in (0, 1/20)$.

Here $m_{\mathcal{H}}$ is defined to be the best possible m_L for $\mathcal{H}$.

VC dimension quantifies sample complexity

Theorem 19

Let $\mathcal{H}$ be a set of functions $\mathcal{X} \to \{0,1\}$. Then the following are equivalent.

- *$\mathcal{H}$ is learnable*
- $\mathrm{VCdim}(\mathcal{H}) < \infty$
- $m_{\mathcal{H}}(\epsilon, \delta) = \Theta(\frac{1}{\epsilon^2} \ln \frac{1}{\delta})$
- $\epsilon_{\mathcal{H}}(m, \delta) = \Theta(\sqrt{\frac{1}{m} \ln \frac{1}{\delta}})$
- $\Pi_{\mathcal{H}}(m)$ *is bounded above by a polynomial in m*
- *There is a function $\epsilon_0(m, \delta) = \Theta(\sqrt{\frac{1}{m} \ln \frac{1}{\delta}})$ with*

$$P^m \left\{ \sup_{h \in \mathcal{H}} |\, \mathrm{er}_P(h) - \hat{\mathrm{er}}_z(h)| > \epsilon_0(m, \delta) \right\} < \delta, \quad \text{for every } P.$$

Linear threshold networks

LTNs

We follow Chapter 6 in [AB09].

Definition 20 (Feedforward network)

A feedforward network is described by a directed graph with nodes enumerated in such a way that each node has only incoming arrows from lower nodes. The number of layers is the longest path from input to output.

Definition 21 (Linear threshold network)

A *linear threshold network* (LTN) is a feedforward network with linear threshold units. A linear threshold unit with inputs x, input weights w and threshold θ is

$$\sigma : x \mapsto \begin{cases} 1, & w^\top x - \theta \geq 0 \\ 0, & \text{otherwise.} \end{cases}$$

Upper bound for LTNs

Theorem 22

Let $\mathcal{H}$ be the class of functions computed on real inputs by a LTN with a total of k computational units and W weights and thresholds. Then, for $m \geq W$,

$$\Pi_{\mathcal{H}}(m) \leq \left(\frac{emk}{W}\right)^{W},$$

and hence

$$\mathrm{VCdim}(\mathcal{H}) < 2W \log_2\left(\frac{2k}{\ln 2}\right).$$

Upper bound for LTNs

Theorem 22

Let $\mathcal{H}$ be the class of functions computed on real inputs by a LTN with a total of k computational units and W weights and thresholds. Then, for $m \geq W$,

$$\Pi_{\mathcal{H}}(m) \leq \left(\frac{emk}{W}\right)^W,$$

and hence

$$\mathrm{VCdim}(\mathcal{H}) < 2W \log_2\left(\frac{2k}{\ln 2}\right).$$

As we will see, this upper bound is essentially tight.

Lower bound for two-layer binary LTNs

Theorem 23

Let $\mathcal{H}$ be the class of functions computed by a two-layer LTN, fully connected between adjacent layers, with $n \geq 3$ binary inputs, k computational units in the first layer, and one output unit. Suppose $k \leq 2^{n+1}/(n^2+n+2)$. Then

$$\text{VCdim}(\mathcal{H}) \geq nk + 1 \geq \frac{3}{5} W,$$

where $W = nk + 2k + 1$ is the total number of weights and thresholds.

Lower bound for two-layer binary LTNs

Theorem 23

Let $\mathcal{H}$ be the class of functions computed by a two-layer LTN, fully connected between adjacent layers, with $n \geq 3$ binary inputs, k computational units in the first layer, and one output unit. Suppose $k \leq 2^{n+1}/(n^2+n+2)$. Then

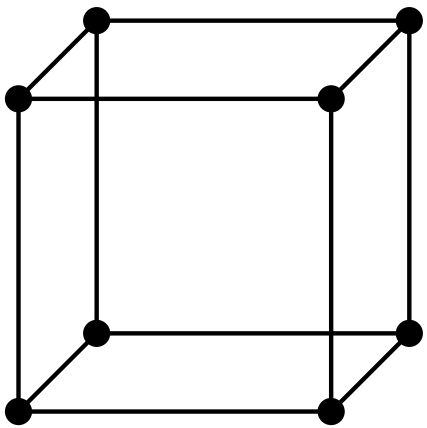
$$\text{VCdim}(\mathcal{H}) \geq nk + 1 \geq \frac{3}{5} W,$$

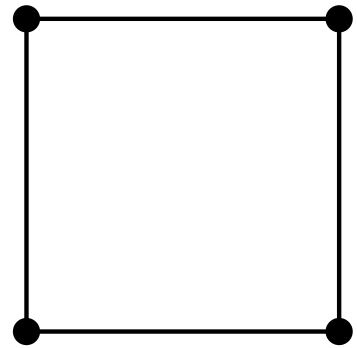
where $W = nk + 2k + 1$ is the total number of weights and thresholds.

This lower bound is only linear in W, but the inputs are binary, and the network has only one hidden layer.

$\{0,1\}^n$

$\{0,1\}^k$





Input space

Hidden units

Figure 3: Binary inputs, separation by hyperplanes.

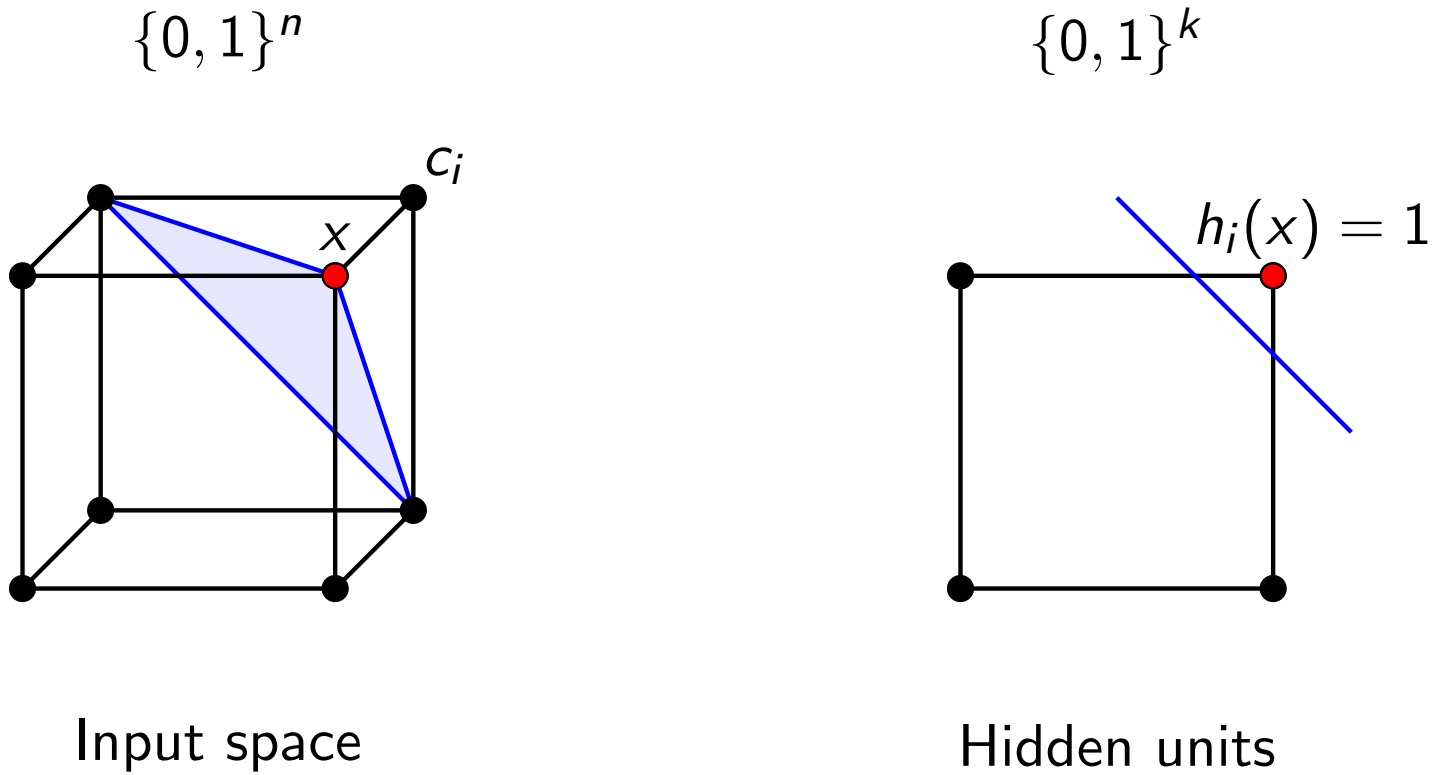


Figure 3: Binary inputs, separation by hyperplanes.

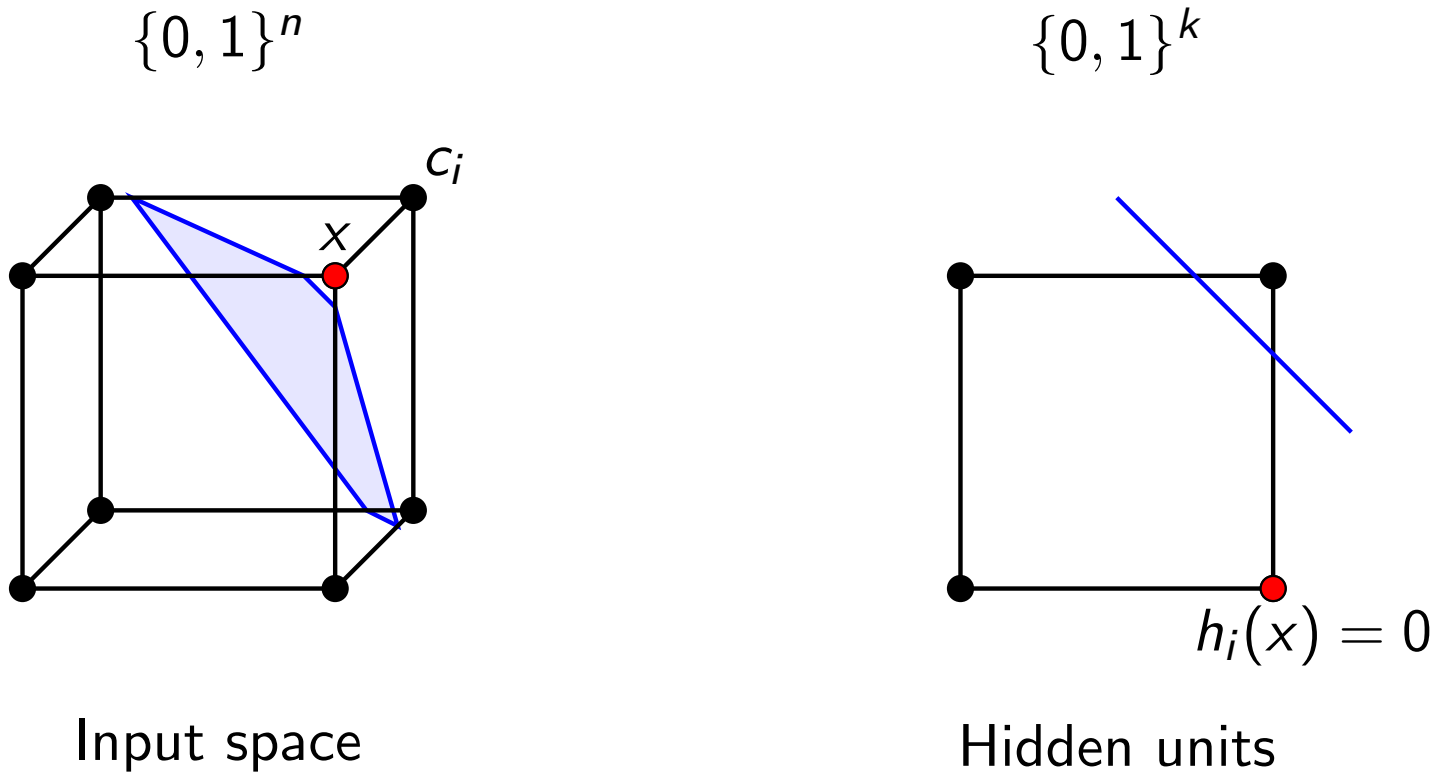


Figure 3: Binary inputs, separation by hyperplanes.

Lower bound for three-layer binary LTNs

We have seen that the VC-dimension of a LTN is bounded above by $O(W \ln W)$, but so far only lower bounds of order W.

The following gives neural nets with superlinear VC-dimension. A single edge can add more than a constant to the VC-dimension.

Theorem 24 ([Maa94])

Let $33 \leq W$. Then there is a three-layer feedforward LTN with W parameters, which computes a function class $\mathcal{H}$ on binary inputs, with

$$\text{VCdim}(\mathcal{H}) > \frac{1}{132} W \log_2 \left(\frac{k}{16} \right),$$

where k is the number of computational units.

Lower bound for two-layer real LTNs

Theorem 25

Consider a two-layer LTN, fully connected between adjacent layers, with $n \geq 3$ real inputs and k computational units, $k \leq 2^{n/2-2}$. Then

$$\text{VCdim}(\mathcal{H}) \geq \frac{nk}{8} \log_2\left(\frac{k}{4}\right) \geq \frac{W}{32} \log_2\left(\frac{k}{4}\right),$$

where $W = nk + 2k + 1$ is the total number of weights and thresholds.

Remarks

- Theorems 24 and 25 give lower bounds of the form $W \log W$. This is within a constant factor of Theorem 22.
- For networks of fixed depth, as W increases the VC-dimension cannot grow $W \log W$ without conditions on the specific architecture.
- For instance, a LTN with layers of size $k_1, k_2, 1$, when $k_2 > 2^{k_1}$ any additional weights in the second layer cannot increase the VC-dimension, since it is already possible to compute all boolean functions of the k_1 first layer units.

Geometric techniques

We follow Chapter 7 in [AB09].

As a generalization of LTFs we discuss binary valued functions obtained by thresholding parametrized real valued functions.

Definition 26

Let $\mathcal{F}$ be a class of real-valued functions defined on $\mathbb{R}^d \times \mathcal{X}$. We say that $\mathcal{H}$ is a k combination of $\operatorname{sgn}(\mathcal{F})$ if there is a function $g\colon \{0,1\}^k \to \{0,1\}$ and functions $f_1, \dots, f_k$ in $\mathcal{F}$ such that for all $h \in \mathcal{H}$ there is a parameter $a \in \mathbb{R}^d$ such that

$$h(x) = g(\operatorname{sgn}(f_1(a,x)), \dots, \operatorname{sgn}(f_k(a,x))), \qquad \forall x \in \mathcal{X}.$$

Now we make a few technical definitions to then formulate and prove an upper bound on the VC-dimension of such classes.

Given a function $f_i : \mathbb{R}^d \to \mathbb{R}$ we denote its zero set by

$$Z_i := \{a \in \mathbb{R}^d : f_i(a) = 0\}.$$

Definition 27

A set $\{f_1, \dots, f_k\}$ of differentiable functions $\mathbb{R}^d \to \mathbb{R}$ has *regular zero-set intersections* if, for all $\{i_1, \dots, i_l\} \subseteq \{1, \dots, k\}$, the Jacobian of $(f_{i_1}, \dots, f_{i_l})$ has rank l at every point of

$$\bigcap_{j=1}^{l} Z_{i_j}.$$

This means that the gradients of these functions are linearly independent at common zeros.

In particular, each function has a non-vanishing gradient at its zeros. Any $d+1$ functions have empty intersection of zero sets.

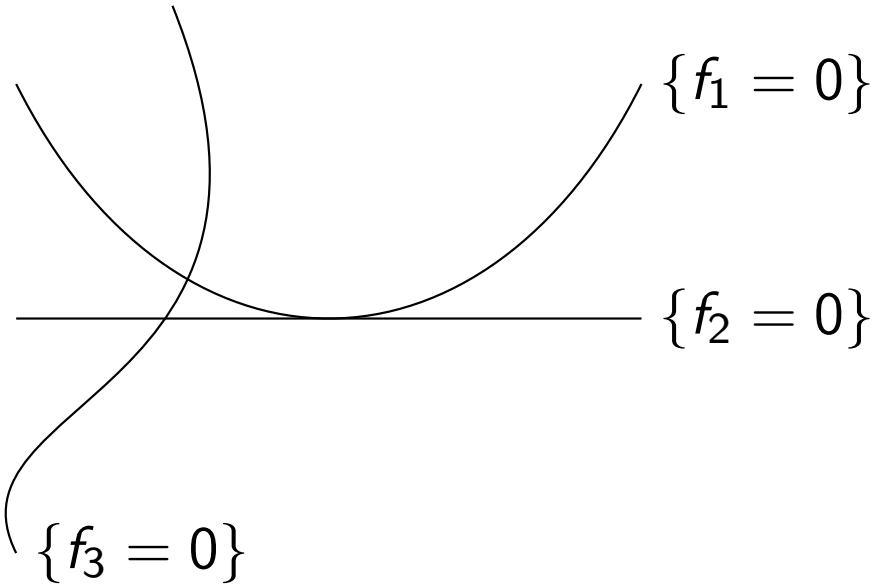


Figure 4: Zero-set intersections for $f_i : \mathbb{R}^2 \to \mathbb{R}$, $i = 1, 2, 3$. The intersection of Z_1 and Z_2 is not regular.

Definition 28

Let G be a set of functions $\mathbb{R}^d \to \mathbb{R}$. We say that G has a solution set components bound B if for any $1 \leq k \leq d$ and any $\{f_1, \ldots, f_k\} \subseteq G$ with regular zero-set intersections, we have

$$CC\left(\bigcap_{i=1}^{k} Z_i\right) \leq B.$$

We will be looking at classes $\mathcal{F}$ of functions $\mathbb{R}^d \times \mathcal{X} \to \mathbb{R}$ with the solution set components bound evaluated for $G = \{a \mapsto f(a, x) \colon f \in \mathcal{F}, x \in \mathcal{X}\}$.

Example 29

The simple perceptron on $\mathbb{R}^d$ corresponds to a $k = 1$ combination of $\mathrm{sgn}(\mathcal{F})$ with $\mathcal{F}$ consisting of functions

$$f(a, x) = \sum_{i=1}^{d} a_i x_i + a_0 + c,$$

with parameter space $\mathbb{R}^{d+1}$. Each of these functions has a hyperplane as zero set. The solution set components bound is $B = 1$.

The most important result in this section.

Theorem 30 (Bound on the growth function)

Let $\mathcal{F}$ be a class of functions $\mathbb{R}^d \times \mathcal{X} \to \mathbb{R}$ that are C^d on $\mathbb{R}^d$, which is closed under addition of constants, and has a solution set components bound B. If $\mathcal{H}$ is a k-combination of $\operatorname{sgn}(\mathcal{F})$, then

$$\Pi_{\mathcal{H}}(m) \leq B \sum_{i=0}^{d} \binom{mk}{i} \leq B \left(\frac{emk}{d} \right)^d ,$$

for $m \geq d/k$.

The proof involves a few steps that we discuss in turn.

First we relate the growth function to the number of regions defined on parameter space by an arrangement of zero sets.

Lemma 31 (Growth function and connected components)

Let $\mathcal{H}$ be a k-combination of $\operatorname{sgn}(\mathcal{F})$, *where $\mathcal{F}$ is a class of functions $\mathbb{R}^d \times \mathcal{X} \to \mathbb{R}$ continuous in the parameters, which is closed under addition of constants. Then there are some $f_1, \ldots, f_k \in \mathcal{F}$ and $x_1, \ldots, x_m \in \mathcal{X}$ for which*

$$\{a \mapsto f_i(a, x_j) \colon i = 1, \ldots, k, j = 1, \ldots, m\}$$

has regular zero-set intersections and

$$\Pi_{\mathcal{H}}(m) \leq CC\left(\mathbb{R}^d \setminus \bigcup_{i=1}^{k} \bigcup_{j=1}^{m} \{a \in \mathbb{R}^d \colon f_i(a, x_j) = 0\}\right).$$

Now we bound the number of regions defined by an arrangement of zero sets, in terms of the number of connected components of intersections of zero sets.

Lemma 32 (Bounding the number of connected components)

Let $\{f_1, \dots, f_k\}$ be a set of differentiable functions $\mathbb{R}^d \to \mathbb{R}$ with regular zero-set intersections. Then

$$CC\left(\mathbb{R}^d \setminus \bigcup_{i=1}^{k} Z_i\right) \leq \sum_{S \subseteq \{1,\dots,k\}} CC\left(\bigcap_{i \in S} Z_i\right).$$

Here we use the convention $\cap_{i \in \emptyset} Z_i = \mathbb{R}^d$.

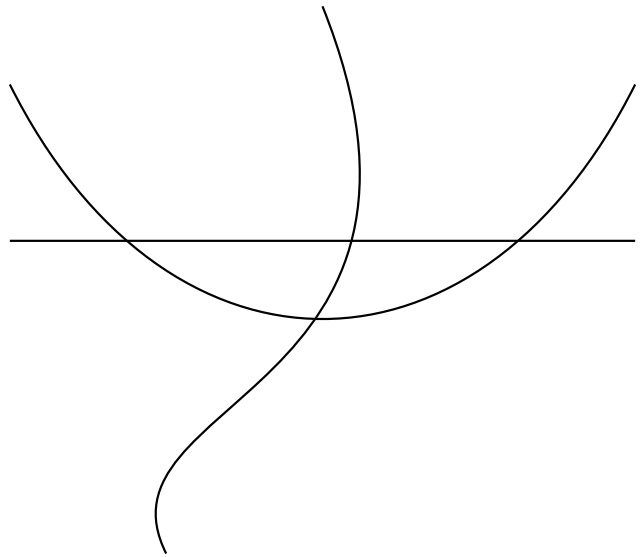


Figure 5: The number of regions is $8 \leq 1 + (1 + 1 + 1) + (2 + 1 + 1)$.

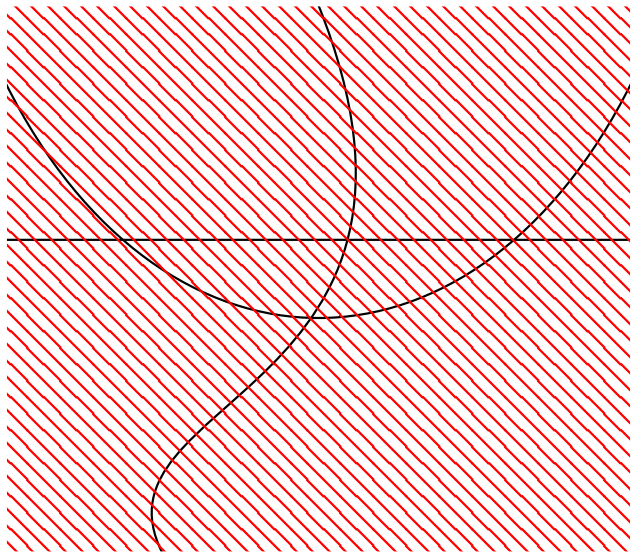


Figure 5: The number of regions is $8 \leq 1 + (1 + 1 + 1) + (2 + 1 + 1)$.

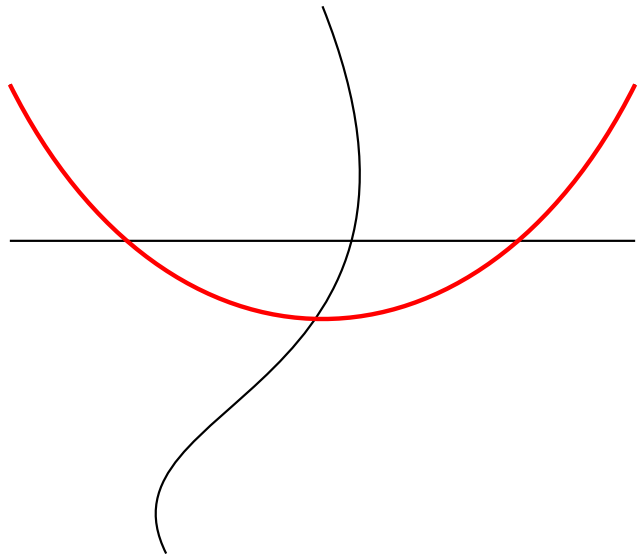


Figure 5: The number of regions is $8 \leq 1 + (1 + 1 + 1) + (2 + 1 + 1)$.

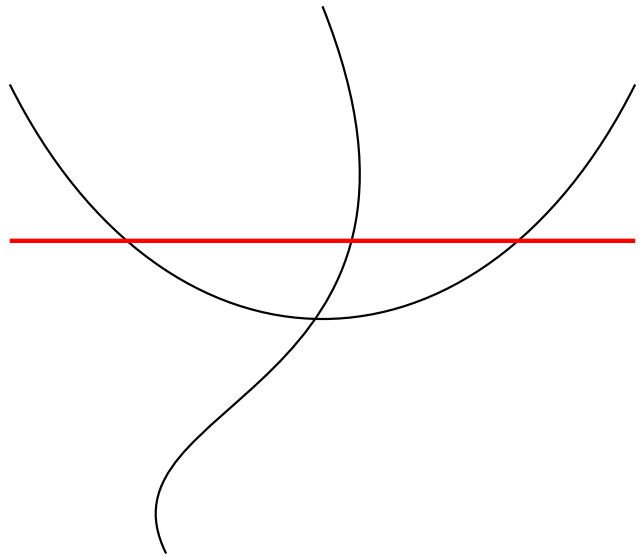


Figure 5: The number of regions is $8 \leq 1 + (1 + 1 + 1) + (2 + 1 + 1)$.

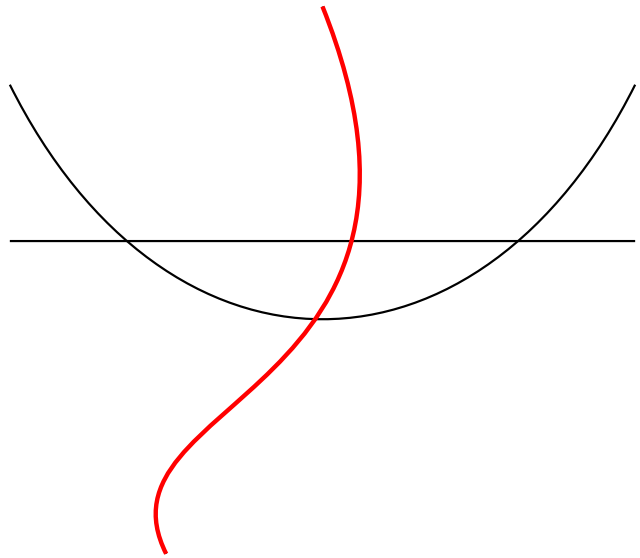


Figure 5: The number of regions is $8 \leq 1 + (1 + 1 + 1) + (2 + 1 + 1)$.

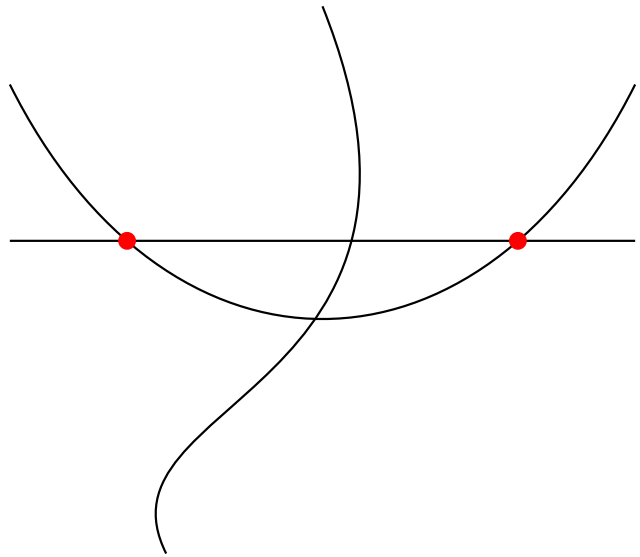


Figure 5: The number of regions is $8 \leq 1 + (1 + 1 + 1) + (2 + 1 + 1)$.

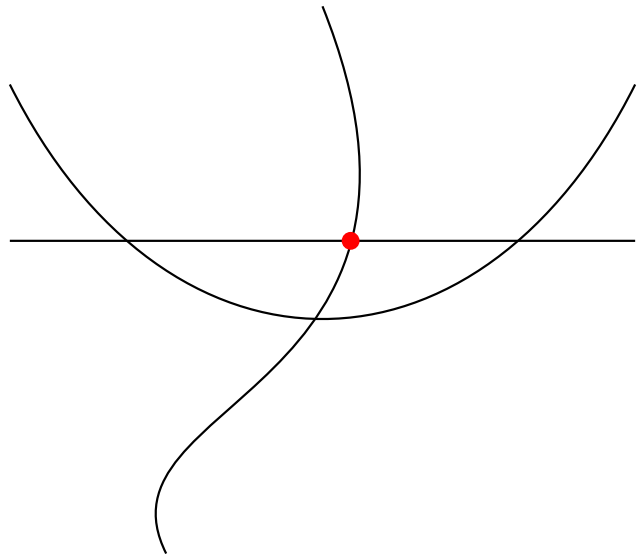


Figure 5: The number of regions is $8 \leq 1 + (1 + 1 + 1) + (2 + 1 + 1)$.

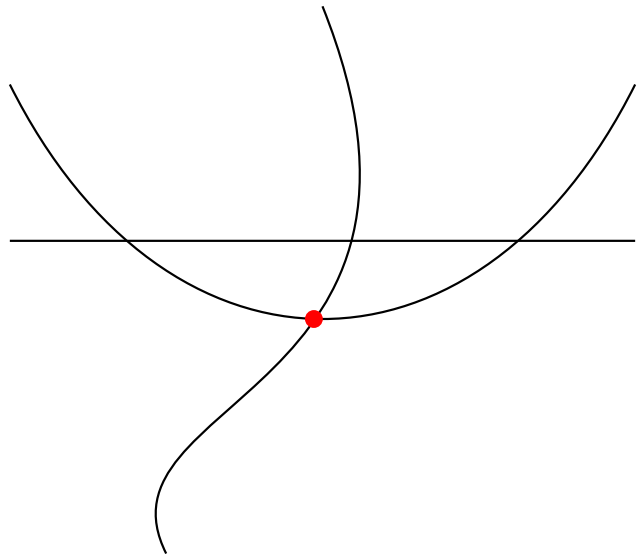


Figure 5: The number of regions is $8 \leq 1 + (1 + 1 + 1) + (2 + 1 + 1)$.

The proof of Lemma 32 uses two lemmas:

Lemma 33

Let $\{f_1, \dots, f_k\}$ and define $S_1, \dots, S_{k-1}$ so that either $S_i = \{a \colon f_i(a) = 0\}$ or $S_i = \{a \colon f_i(a) \neq 0\}$.
Let C be a connected component of $\cap_{i=1}^{k-1} S_i$, and C' a connected component of $C \cap \{a \colon f_k(a) = 0\}$. Then $C \setminus C'$ has at most two connected components.

This is saying that removing a connected set from another connected set leaves at most two connected components.

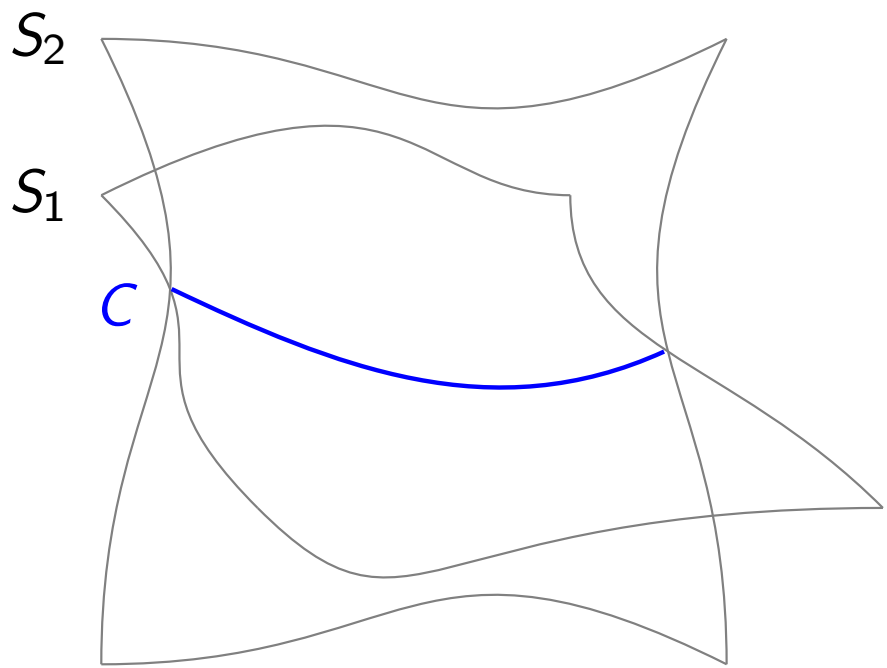


Figure 6: Removing from C a connected component C' of its intersection $C \cap S_k$ with a zero set S_k leaves a set $C \setminus C'$ that has at most two connected components.

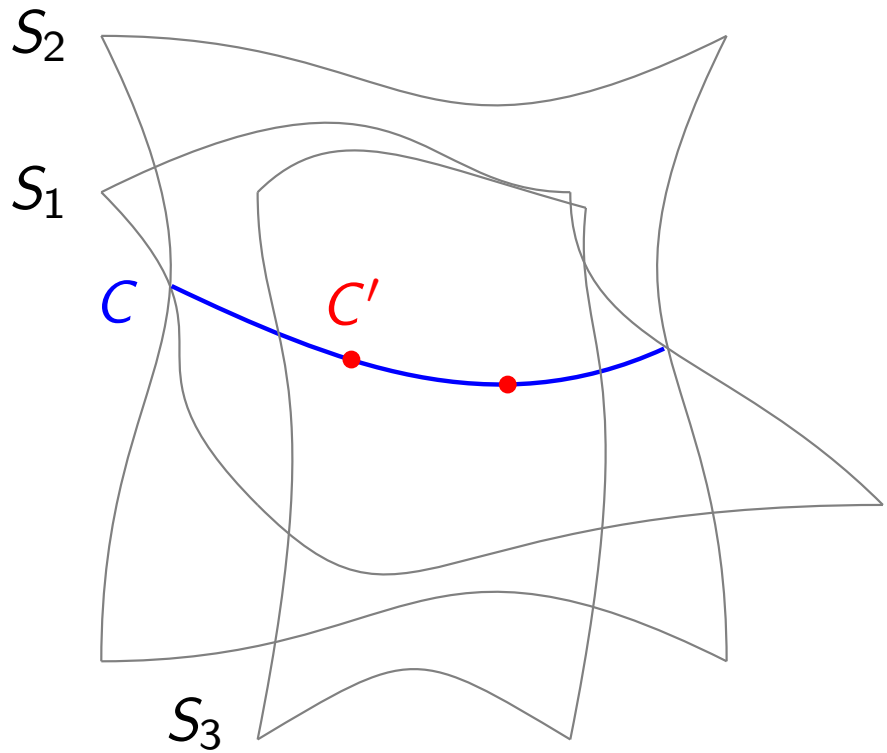


Figure 6: Removing from C a connected component C' of its intersection $C \cap S_k$ with a zero set S_k leaves a set $C \setminus C'$ that has at most two connected components.

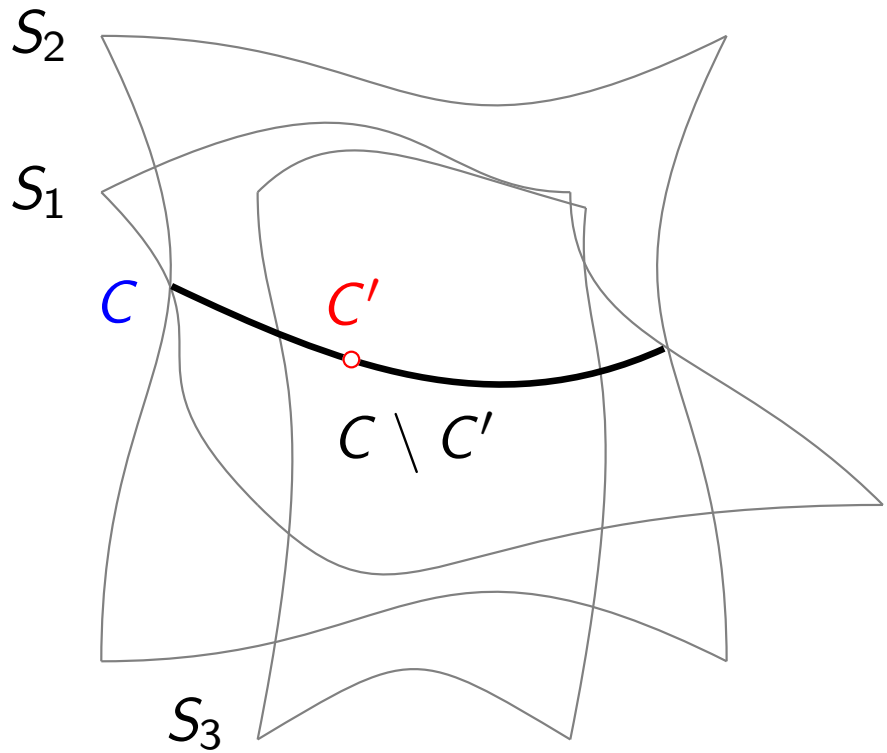


Figure 6: Removing from C a connected component C' of its intersection $C \cap S_k$ with a zero set S_k leaves a set $C \setminus C'$ that has at most two connected components.

The second lemma gives an induction on the number of regions, similar to the sweep hyperplane method:

Lemma 34

Consider $Z_1, \dots, Z_k$ *and* $I \subseteq \{1, \dots, k\}$. *Let* $M = \cap_{i \in I} Z_i$ *and* $\{M_1, \dots, M_b\} = \{Z_i : i \neq I\}$, *where* $b = k - |I|$. *Then*

$$CC(M \setminus \cup_{j=1}^{b} M_j) \leq CC(M \setminus \cup_{j=1}^{b-1} M_j) + CC((M \cap M_b) \setminus \cup_{j=1}^{b-1} M_j).$$

We are discussing the number of regions into which a set M is split by an arrangement of zero sets $M_1, \dots, M_b$.

Adding M_b to the arrangement $M_1, \dots, M_{b-1}$ adds at most as many regions as are defined by $M_1, \dots, M_{b-1}$ on $M \cap M_b$.

General bounds for neural networks

We follow [AB09, Chapter 8].

We consider first classes of functions that are polynomial in their parameters. For these we can obtain a solution set component bound B as follows.

Theorem 35 (Milnor-Thom Theorem, see [Wal96])

Let $V \subseteq \mathbb{R}^d$ be the set of common zeros of polynomials f_i, $i = 1, \ldots, p$, each of degree at most l. Then the sum of Betti numbers is bounded above as

$$b_0(V) + \cdots + b_d(V) \leq l(2l-1)^{d-1}.$$

In particular, the number of connected components satisfies

$$CC(V) = b_0(V) \leq l(2l-1)^{d-1} \leq (2l)^d.$$

For k-combinations from a polynomially parametrized class we can bound the growth function and VC-dimension as follows:

Theorem 36

Let $\mathcal{F}$ be a class of functions $\mathbb{R}^d \times \mathcal{X} \to \mathbb{R}$ so that, for all $x \in \mathcal{X}$ and $f \in \mathcal{F}$, the function $a \mapsto f(a, x)$ is a polynomial of degree $\leq l$. Let $\mathcal{H}$ be a k-combination of $\operatorname{sgn}(\mathcal{F})$. Then

$$\Pi_{\mathcal{H}}(m) \leq \left(\frac{2emkl}{d}\right)^d,$$

whenever $m \geq d/k$. In particular,

$$\operatorname{VCdim}(\mathcal{H}) \leq 2d \log_2(12kl).$$

Theorem 37

Let h be a function $\mathbb{R}^d \times \mathbb{R}^n \to \{0,1\}$ and

$$\mathcal{H} = \{x \mapsto h(a,x) \colon a \in \mathbb{R}^d\}.$$

Assume that h can be computed by an algorithm that takes as input (a,x) and returns $h(a,x)$ after no more than t arithmetic operations $+,-,\times,/$ on real numbers, and jumps conditioned on $<,>,\leq,\geq,=,\neq$ comparisons of real numbers. Then

$$\text{VCdim}(\mathcal{H}) \leq 4d(t+2).$$

Proof sketch.

- Denote A the algorithm that computes h.
- Any comparison performed by A can be expressed as a comparison of two polynomials of degree at most 2^t.
- There are at most 2^t distinct polynomials examined by A.
- Hence, by Theorem 36
  $\text{VCdim}(\mathcal{H}) \leq 2d \log_2(12 \cdot 2^t 2^t) \leq 2d(2t + \log_2(12))$.

Remarks

- Theorem 37 implies that if a model has finitely many parameters, only allows standard arithmetic operations on reals, and can be computed in finite time, then it has a finite VC-dimension.
- If the number of parameters and computation time grows only polynomially with the input space dimension n, then the VC-dimension also grows only polynomially with n. In turn (Lec 2) the sample complexity also grows only polynomially.
- Interestingly, the result no longer holds if one includes operations such as $\lfloor \cdot \rfloor$ or $\sin(\cdot)$ (in unit time). In fact, such functions can be used to define function classes with infinite VC-dimension.

The previous Theorem 37 is optimal up to constant factors:

Theorem 38

For all $d, t \geq 1$, there is a class of functions, $\mathcal{H}$, parametrized by d real values, that can be computed in time $O(t)$, and that has $\text{VCdim}(\mathcal{H}) \geq dt$.

Note that this is an existence statement which does not necessarily hold for a given model with d parameters computable in $O(t)$.

We use the previous Theorem 37 to derive a result for networks with piecewise polynomial activation functions.

Theorem 39 (Networks with piecewise polynomial activations)

Let N be a feedforward network with W weights and k computation units. Assume that each internal unit has a piecewise-polynomial activation function with p pieces and degree no more than l, and the output unit is a linear threshold unit. Then, for the class of functions $\mathcal{H}$ computed by N we have

$$\text{VCdim}(\mathcal{H}) = O(W(W + kl \log_2(p))).$$

Here $f(\vec{x}) = O(g(\vec{x}))$ if and only if there are constants $M, C > 0$ such that for all $\vec{x}$ with $x_i \geq M$ for all i, $|f(\vec{x})| \leq C|g(\vec{x})|$.

The previous Theorem 39 can be improved as follows.

Theorem 40

Consider a feedforward network with L layers, W weights, k computation units, all internal units having piecewise polynomial activations with p pieces and degree l. Then

$$\Pi_{\mathcal{H}}(m) \leq \prod_{i=1}^{L} \left(\frac{2emk_i p(l+1)^{i-1}}{W_i} \right)^{W_i}$$

Where k_i is the number of units in layer i and W_i is the number of weights involved in computing the activations in layer i.
For fixed p and l,

$$\text{VCdim}(\mathcal{H}) = O(W(L \log_2(W) + L^2)).$$

The previous two results are nearly optimal:

Theorem 41

Let $s : \mathbb{R} \to \mathbb{R}$ *satisfy*

(i) $\lim_{\alpha \to -\infty} s(\alpha) = 0$ *and* $\lim_{\alpha \to \infty} s(\alpha) = 1$, *and*

(ii) s *is differentiable with* $s'(\alpha_0) \neq 0$ *at some* α_0.

Then, for any $L \geq 1$ *and* $W \geq 10L - 14$, *there is a feedforward network with* L *layers,* W *parameters, computation units with activation* s *(except the output unit, which is a LT unit), for which*

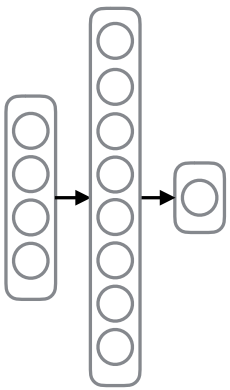
$$\text{VCdim}(\mathcal{H}) \geq \left\lfloor \frac{L}{2} \right\rfloor \left\lfloor \frac{W}{2} \right\rfloor .$$

The proof strategy is similar to that of Theorem 38.

Shallow or Deep?

- VC analysis indicated following behavior for shallow and deep networks, for a variety of activation functions

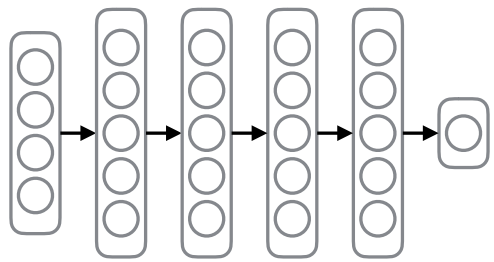
Shallow



$$\mathrm{er}_P(h) \sim \mathrm{opt}_P(\mathcal{S}) + \sqrt{\frac{W \log(W)}{N}}$$

Error Approximation Estimation

Deep



$$\mathrm{er}_P(h) \sim \mathrm{opt}_P(\mathcal{D}) + \sqrt{\frac{W^2}{N}}$$

Error Approximation Estimation

- On the other hand, in numerous experiments deep networks tend to work better than shallow networks
- We conclude (leaving aside parametrization, optimization) that certain functions relevant in practice are expressed much more compactly in terms of deep networks.

1. Observations
   - The curse of dimensionality
   - Multivariate non-linear functions
   - Is learning feasible?

2. Supervised Learning Theory
   - A formal definition of learning
   - Growth function and VC dimension
   - Linear threshold networks
   - Geometric techniques
   - General bounds for neural networks

3. Approximation Properties
   - Universal approximation and rate of convergence
   - Deep and Shallow
   - Combinatorial views on ReLUs

Universal approximation and rate of convergence

A central result on approximation of continuous functions:

Theorem 42 (Weierstrass)

Suppose f is a continuous real-valued function defined on the real interval $[a, b]$. For each $\epsilon > 0$, there is a polynomial p such that for all $x \in [a, b]$, we have $|f(x) - p(x)| < \epsilon$.

More generally:

Theorem 43 (Stone-Weierstrass)

Suppose $\mathcal{X}$ is a compact Hausdorff space and A is a subalgebra of $C(\mathcal{X}, \mathbb{R})$ which contains a non-zero constant function. Then A is dense in $C(\mathcal{X}, \mathbb{R})$ if and only if it separates points.

There is also a version for locally compact spaces.

We follow [RH95, Chapter 5.7].

- A logistic unit is $\sigma : s \mapsto \frac{1}{1+\exp(-s)}$
- A threshold unit is $\sigma : s \mapsto \begin{cases} 0, & s < 0 \\ 1, & s \geq 0 \end{cases}$
- A ramp unit is $\sigma : s \mapsto \max\{0, \min\{s, 1\}\}$
- A rectified linear unit is $\sigma : s \mapsto \max\{0, s\}$

Universal approximation theorem

Theorem 44

Any continuous function $f : \mathbb{R}^n \to \mathbb{R}^p$ can be approximated uniformly on compacta by functions of the form

$$y_k = f_k\left(\alpha_k + \sum_j w_{jk} f_j\left(\alpha_j + \sum_{i=1}^{n} w_{ij} x_i\right)\right), \quad k = 1, \ldots, p,$$

with linear output units and logistic units in the hidden layer, and also by networks with threshold units, ramp units, rectified linear units in the hidden layer.

Rate of convergence

For the rate of convergence a well known result is:

Theorem 45 ([Bar93])

Suppose $f : \mathbb{R}^n \to \mathbb{R}$ has a Fourier representation of the form

$$f(x) = \int_{\mathbb{R}^n} e^{i\omega^\top x} \tilde{f}(\omega) d\omega, \quad \text{with} \quad C_f = \int_{\mathbb{R}^n} \|\omega\| |\tilde{f}(\omega)| d\omega < \infty.$$

Then for each N, f can be approximated by a function of the form

$$\alpha_o + \sum_{j=1}^{N} w_{jo} f_j \left(\alpha_j + \sum_{i=1}^{n} w_{ij} x_i \right)$$

with N hidden units (logistic, threshold, or ramp), in L_2 on $B_r = \{\|x\| < r\}$ with error at most $(2rC_f)/\sqrt{N}$.
Further, we can take $\alpha_o = f(0)$ and $\sum_j |w_{jo}| \leq 2rC_f$ for the hidden-to-output weights.

Remark 1

- Barron's theorem has been regarded as surprising in that the rate of convergence seems to be independent of n. The typical rate of convergence in approximation theorems is $O(N^{-c/n})$, where c depends on how smooth f is.
- However, the conditions do depend on n.
  - First, B_r becomes much smaller as n increases; the radius of a ball containing the unit hypercube is $\sqrt{n}$.
  - Second, the integral for C_f is dimension dependent; considering radially symmetric functions $f(x) = f(\|x\|)$ suggests that normally C_f grows exponentially fast with n.
- The condition $C_f < \infty$ implies that f is continuously differentiable with a gradient whose Fourier transform is integrable. This is satisfied if f has $\lfloor n/2 \rfloor + 2$ continuous derivatives.

Remark 2

- To approximate all functions with r bounded derivatives to accuracy $1/N$ in L_2, results by DeVore indicate that at least $\Omega(N^{n/r})$ units are needed.
- So the *curse of dimensionality* cannot be broken by neural networks (nor by any similar method).

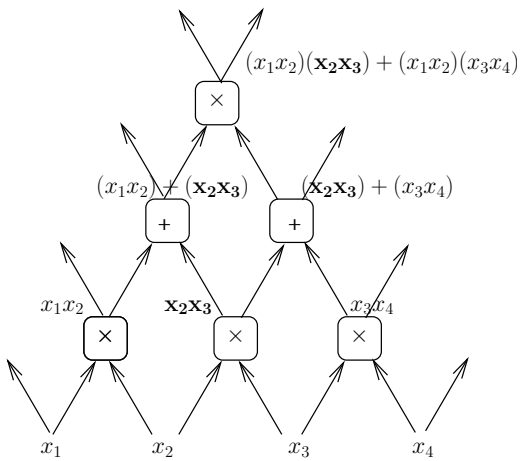
Deep and Shallow

Intuition

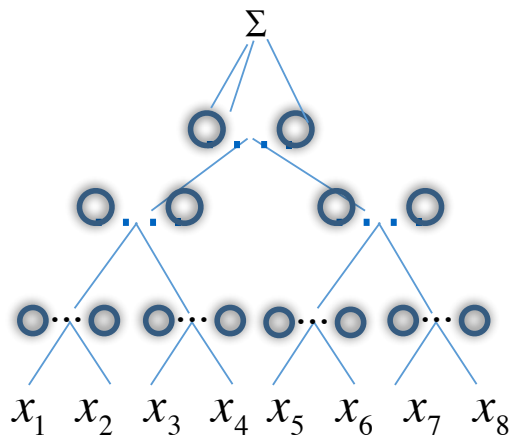
- How we arrange computations matters

$$p(x) = a_0 + a_1 x + \cdots + a_n x^n$$
$$= a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + a_n x)))$$

- Certain functions are expressed more compactly in terms of deep networks.



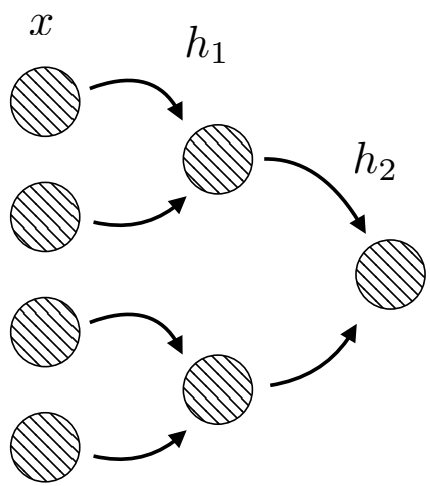
[Bengio '09]



[Poggio et al '16]

Intuition

- Each layer can map several inputs to the same output



765	765	765	765	765
765	765	751	548	346
765	722	333	225	266
764	416	209	232	279
765	313	212	239	286
764	480	219	252	306
765	656	408	353	687

We follow [Ben09].

Deep vs. shallow

- If a function can be compactly represented with k levels using a particular set of computational elements, it might require a huge number of computational elements to represent it with $k-1$ levels (using the same computational elements).
- In theory (parametrization and optimization aspects aside), the number of computational elements one can afford depends on the number of training examples available.
- The consequences are not just computational but also statistical: poor generalization may be expected when using an insufficiently deep architecture for representing some functions.

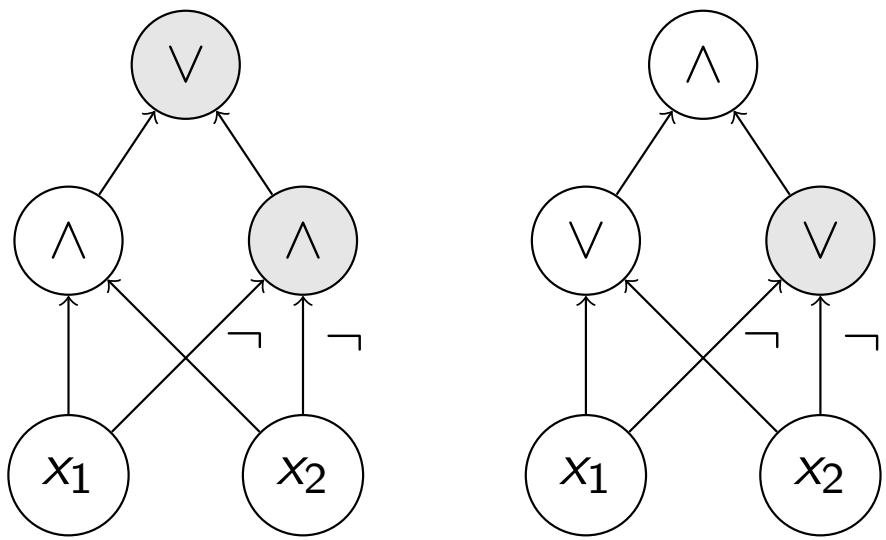
The polynomial $\prod_{i=1}^{n} \sum_{j=1}^{m} a_{ij} x_j$ can be represented efficiently as a product of sums, with only $O(mn)$ operations.

On the other hand, it would be very inefficiently represented as a sum of products, requiring $O(m^n)$ operations.

Different network architectures might correspond to more or less efficient types of factorization, depending on the type of function that one wants to represent.

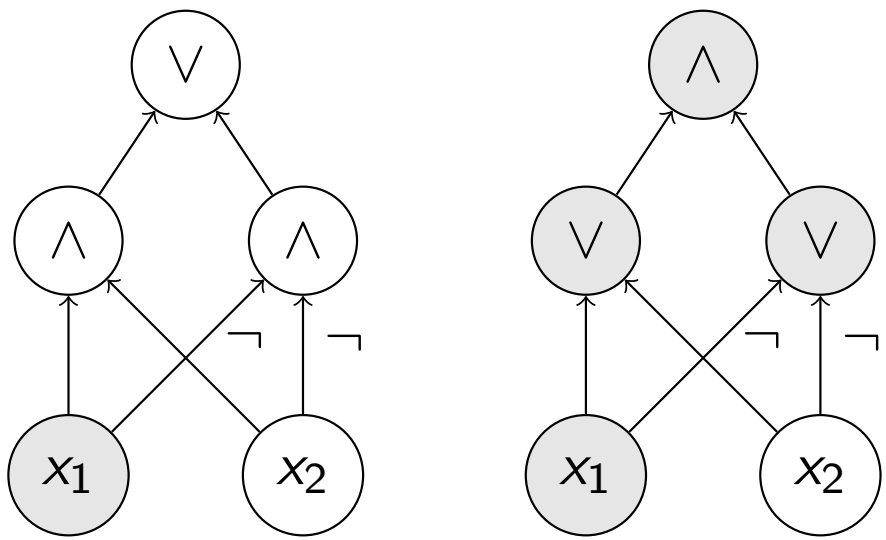
Let's have a look at logic circuits first:

- A two-layer circuit of logic gates can represent any boolean function. Any boolean function can be written as
  - Sum of products
    (Disjunctive Normal Form: AND gates on the first layer with optional negation of inputs, and OR gate on the second layer) or
  - Product of sums
    (Conjunctive Normal Form: OR gates on the first layer with optional negation of inputs, and AND gate on the second layer).



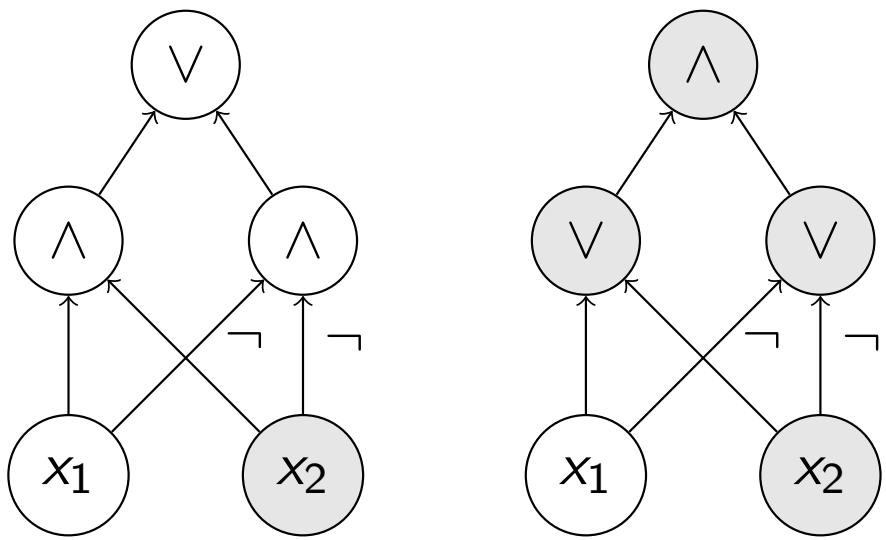
Let's have a look at logic circuits first:

- A two-layer circuit of logic gates can represent any boolean function. Any boolean function can be written as
  - Sum of products
    (Disjunctive Normal Form: AND gates on the first layer with optional negation of inputs, and OR gate on the second layer) or
  - Product of sums
    (Conjunctive Normal Form: OR gates on the first layer with optional negation of inputs, and AND gate on the second layer).



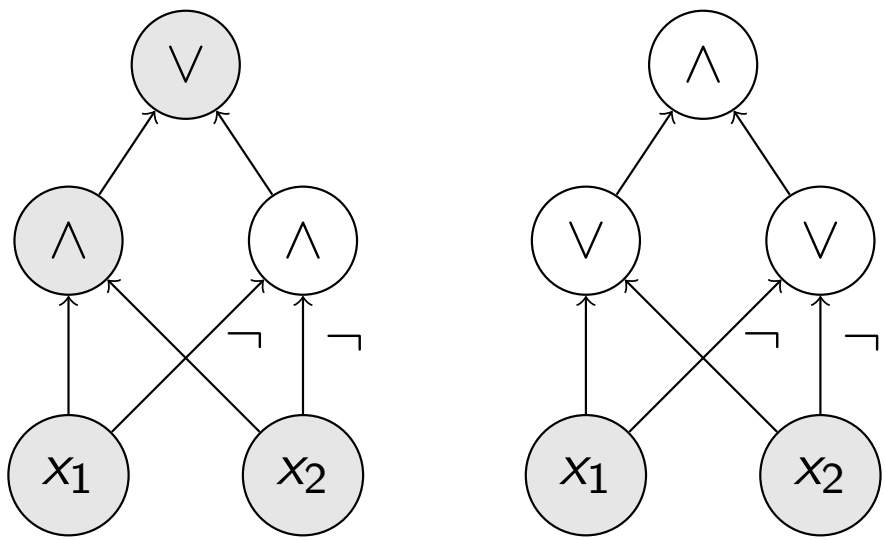
Let's have a look at logic circuits first:

- A two-layer circuit of logic gates can represent any boolean function. Any boolean function can be written as
  - Sum of products
    (Disjunctive Normal Form: AND gates on the first layer with optional negation of inputs, and OR gate on the second layer) or
  - Product of sums
    (Conjunctive Normal Form: OR gates on the first layer with optional negation of inputs, and AND gate on the second layer).



Let's have a look at logic circuits first:

- A two-layer circuit of logic gates can represent any boolean function. Any boolean function can be written as
  - Sum of products
    (Disjunctive Normal Form: AND gates on the first layer with optional negation of inputs, and OR gate on the second layer) or
  - Product of sums
    (Conjunctive Normal Form: OR gates on the first layer with optional negation of inputs, and AND gate on the second layer).



- However, there are functions computable with a polynomial-size logic gates circuit of depth k that require exponential size when restricted to depth $k-1$.
- Any d-bit parity circuits of depth 2 have exponential size. The d-bit parity function is defined as

$$(b_1, \ldots, b_d) \in \{0,1\}^d \mapsto \begin{cases} 1, & \text{if } \sum_{i=1}^d b_i \text{ is even,} \\ -1, & \text{otherwise.} \end{cases}$$

Lemma 46

Depth 2 circuits computing parity are of size 2^{n-1}. The circuit can be written either as an AND of ORs or an OR of ANDs.

Proof.

- By De Morgan's laws, $\neg(P \vee Q) \Leftrightarrow (\neg P) \wedge (\neg Q)$, and $\neg(P \wedge Q) \Leftrightarrow (\neg P) \vee (\neg Q)$, one can restrict attention to circuits where negation only occurs at the input level.
- Further, one can restrict attention to alternating levels of AND and OR, since two adjacent gates of the same type can be collapsed into one gate.
- Consider a circuity with ANDs followed by the OR output. Let σ be an input vector such that parity is 1. Consider the unique AND of n literals which is 1 only for σ. There are precisely 2^{n-1} such ANDs and the OR of them is equal to parity.
- ORs followed by AND is Homework.

Theorem 47 (Hastad 1986)

Parity can be computed by circuits of size $O(n^{\frac{k-2}{k-1}} 2^{n^{1/(k-1)}})$ *and depth* k*. The output gate can be either an AND or an OR.*

With $k = \log(n)$ layers we obtain a network of size $O(n)$.

We have seen exponential gaps between the necessary and sufficient size of shallow and deep networks to compute certain functions.

In terms of universal representation,

Theorem 48 (Shannon 1942)

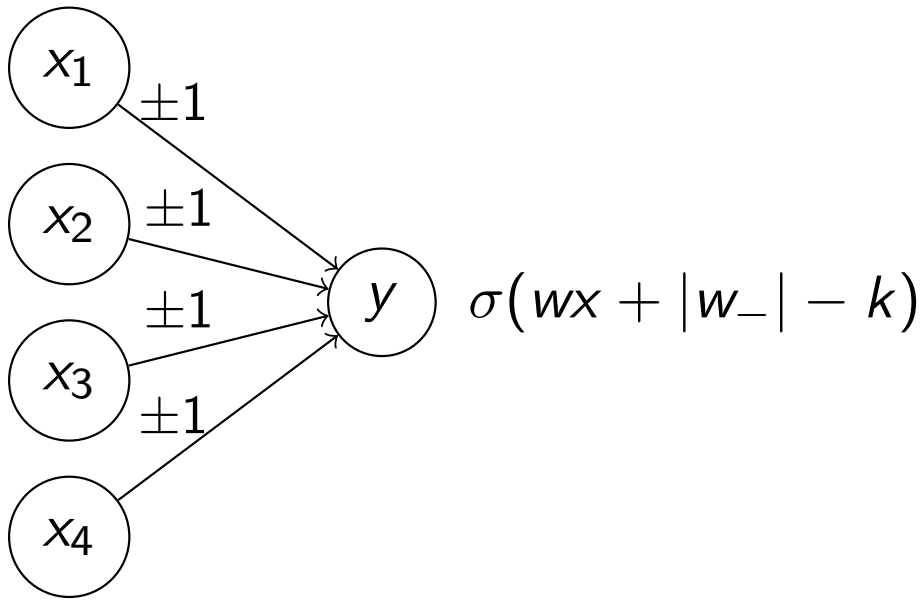
Almost all n-input Boolean functions require a logic circuit of size at least $(1 - o(1))2^n/n$.

Theorem 49 (Lupanov 1952)

Every n-input Boolean function can be expressed by a logic circuit of size at most $(1 + o(1))2^n/n$.

So, the name of the game is to focus on specific classes.

What about other types of computational elements?
A threshold unit generalizes a Boolean unit.



This outputs 1 iff the number of x_i with the desired value, as expressed in w, is at least k.
A logic unit corresponds to $k = 1$ (OR) or $k = n$ (AND).

A monotone weighted threshold circuit is a multilayer network with linear threshold units and positive weights.

There is a class $\mathcal{F}_{k,N}$ containing functions of $N^{2(k-1)}$ variables defined by a depth k circuit of size $N^{2(k-1)}$, for which the following holds.

Theorem 50 ([HG91])

Computing a function $f_k \in \mathcal{F}_{k,N}$ with a monotone weighted threshold circuit of depth $k-1$ requires size at least 2^{cN} for some constant $c > 0$ and $N > N_0$.

Now let us have a look at approximation of functions with other types of activation functions.

We follow [MP16].

- Both shallow and deep networks are universal.
- The approximation of functions with a compositional structure can be achieved with the same degree of accuracy by deep and also by shallow networks.
- However, the number of parameters and VC-dimension are much smaller for the deep networks than for the shallow networks with equivalent approximation accuracy.
- It is intuitive that a hierarchical network matching the structure of the compositional function should be better at approximating it.

- Consider the domain $I^n = [-1, 1]^n$ and norm $\|f\| = \max_{x \in I^n} |f(x)|$.
- For a set of continuous functions $G \subseteq C(I^n)$, define $\text{dist}(f, G) = \inf_{g \in G} \|f - q\|$.

Definition 51 (Shallow functions and smoothness)

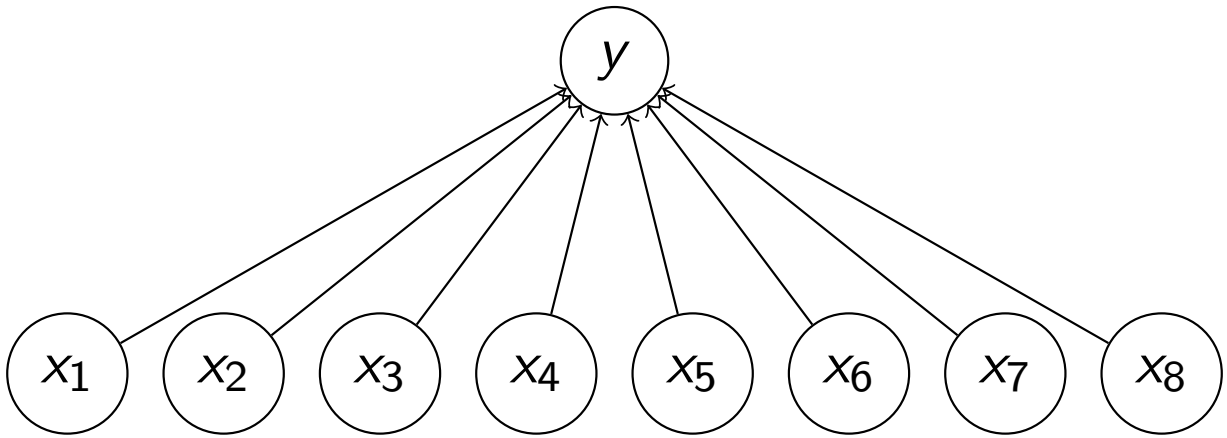
Let $\mathcal{S}_N$ denote the class of functions representable by a shallow network with N units, as

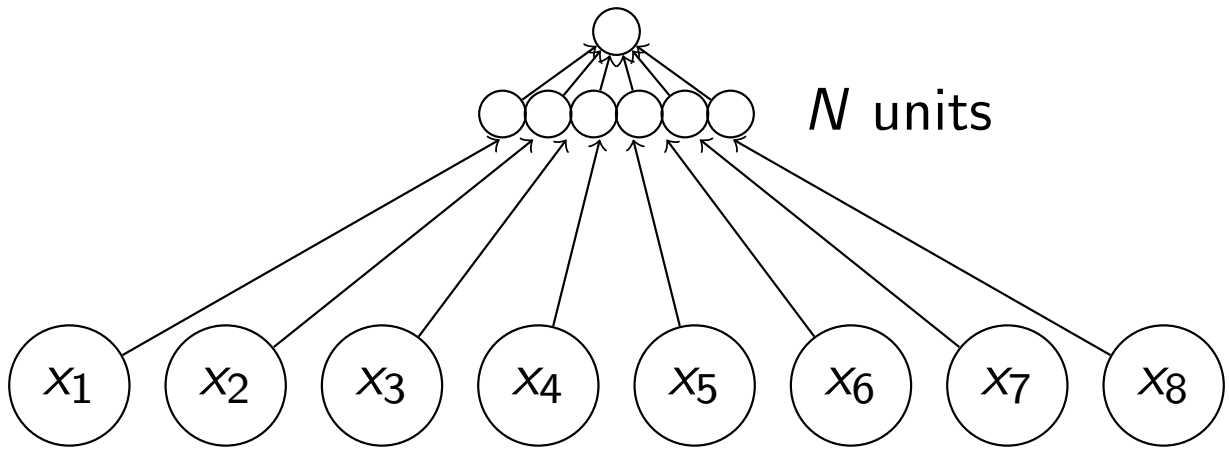
$$x \mapsto \sum_{k=1}^{N} a_k \sigma(w_k^\top x + b_k),$$

where $w_k \in \mathbb{R}^n$, $b_k, a_k \in \mathbb{R}$.

The number of trainable parameters is $nN + N + N = O(nN)$.

Let $W_{r,n}$ be the set of all functions with continuous partial derivatives of orders up to r such that $\sum_{0 \leq |k|_1 \leq r} \|D^k f\| \leq 1$.


y
x1
x2
x3
x4
x5
x6
x7
x8


N units
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8

Definition 52 (Compositional functions and smoothness)

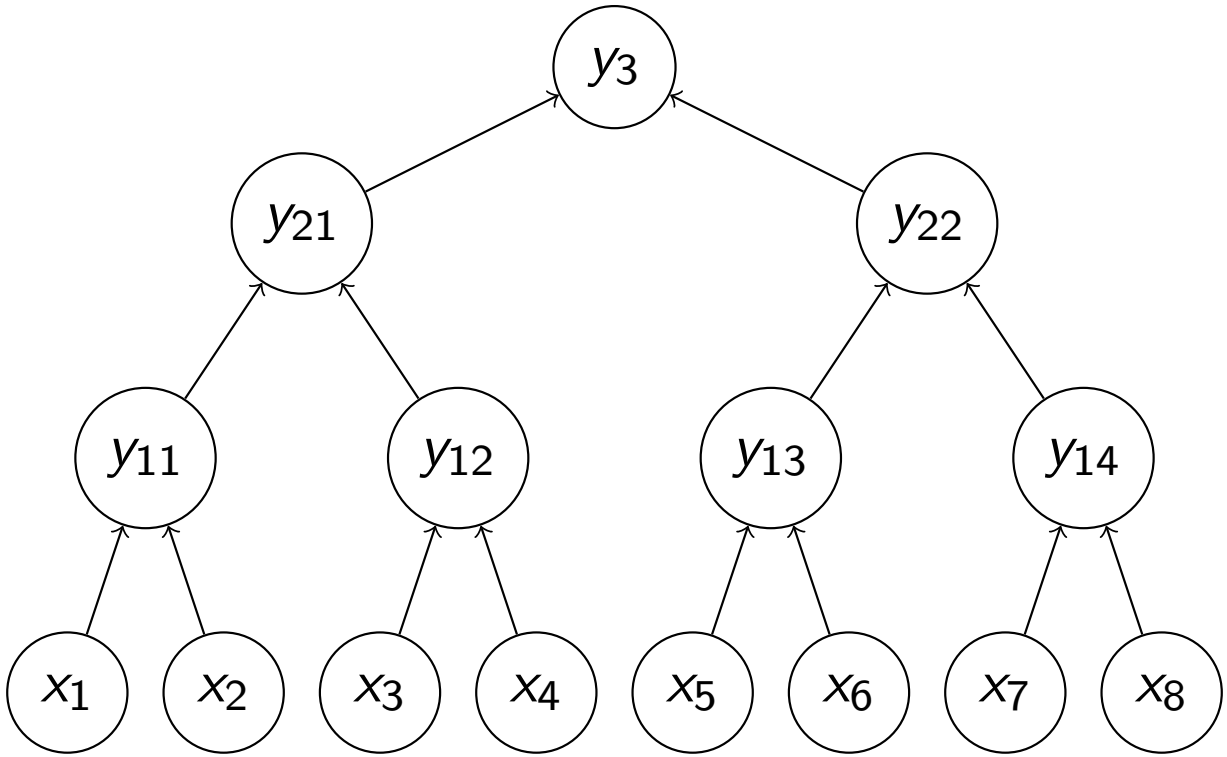
Let $W_{H,r,2}$ be the class of compositional functions with a binary tree structure, such as

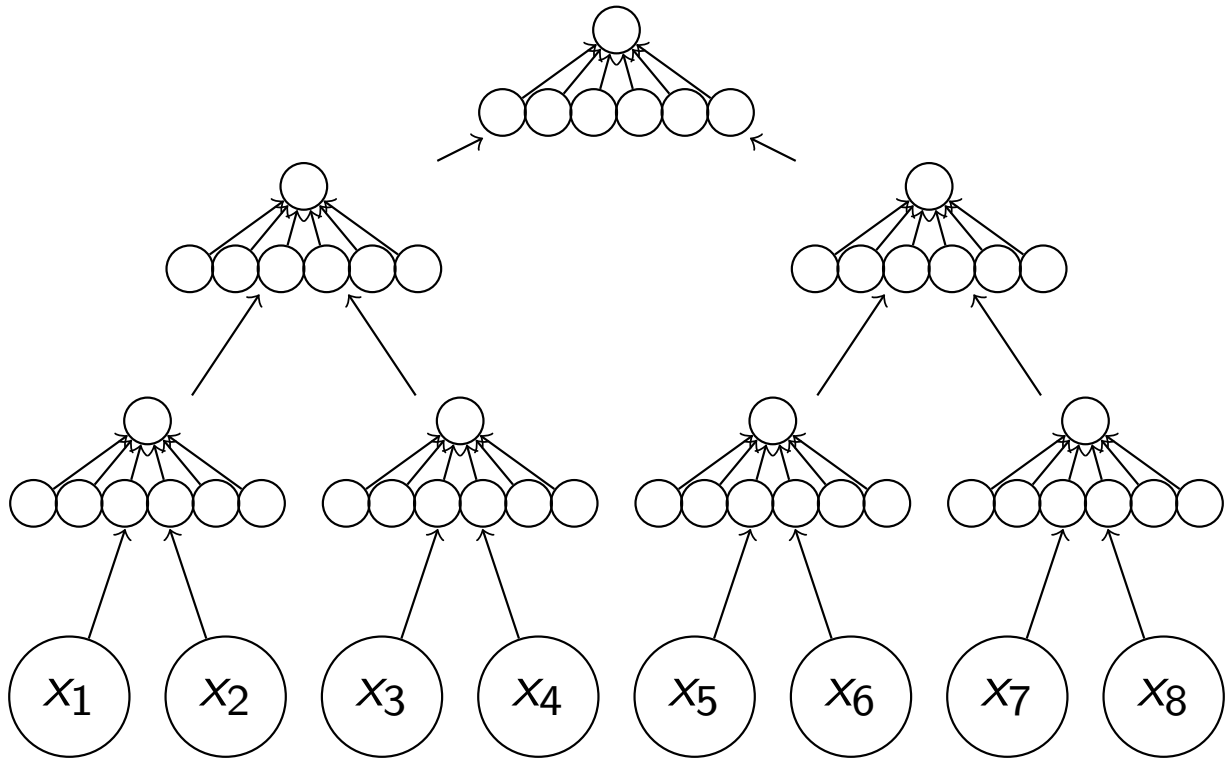
$$f(x_1, \dots, x_8) \\ = h_3(h_{21}(h_{11}(x_1, x_2), h_{12}(x_3, x_4)), h_{22}(h_{13}(x_5, x_6), h_{14}(x_7, x_8))),$$

with constituent functions h in $W_{r,2}$.

Denote $\mathcal{D}_N$ the class of functions representable by a network with the same structure, with constituent functions in $\mathcal{S}_N$.

The number of trainable parameters is
$\sum_l \frac{n}{2^l}(2N + N) + N = O(nN)$.


y_3
y_{21}
y_{22}
y_{11}
y_{12}
y_{13}
y_{14}
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8


x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8

Theorem 53 ([MP16])

Let $\sigma : \mathbb{R} \to \mathbb{R}$ be infinitely differentiable, and not a polynomial on any subinterval of $\mathbb{R}$.

(a) *For $f \in W_{r,n}$, $\text{dist}(f, \mathcal{S}_N) = O(N^{-r/n})$.*

(b) *For $f \in W_{H,r,2}$, $\text{dist}(f, \mathcal{D}_N) = O(N^{-r/2})$.*

Given some family G_N, if $\text{dist}(f, G_N) = O(N^{-\gamma})$ for some $\gamma > 0$, then one needs a network with $N = O(\epsilon^{-1/\gamma})$ to guarantee an approximation to accuracy ϵ.

Hence, to guarantee ϵ accuracy, the number of trainable parameters is $O(\epsilon^{-n/r})$ (shallow) and $O(\epsilon^{-2/r})$ (hierarchical).

Proof. Part (a) is given here without proof.
Part (b). Each constituent function is in $W_{r,2}$. Applying (a) with $n = 2$ implies that each constituent function can be approximated by $\mathcal{S}_N$ up to accuracy $N^{-r/2}$. The assumption $f \in W_{H,r,2}$ implies that each constituent function is Lipschitz continuous (it has bounded derivative). Hence, for example, if P, P_1, P_2 are approximations of h, h_1, h_2 to accuracy ϵ, then

$$\begin{aligned}
&\|h(h_1, h_2) - P(P_1, P_2)\| \\
&\leq \|h(h_1, h_2) - h(P_1, P_2)\| + \|h(P_1, P_2) - P(P_1, P_2)\| \\
&\leq c(\|h_1 - P_1\| + \|h_2 - P_2\| + \|h - P\|) \leq 3c\epsilon
\end{aligned}$$

for some $c > 0$ independent of ϵ. The number of units in the network is of order n. □

- Are these bounds optimal?
- Consider some normed linear space V and a compact subset $W \subseteq V$. The nonlinear n-width of W is

$$d_n(W) = \inf_{A_n, M_n} \sup_{f \in W} \operatorname{dist}(f, A_n(M_n(f))),$$

  where the infimum is over all continuous $M_n \colon W \to \mathbb{R}^n$ and any $A_n \colon \mathbb{R}^n \to V$.
- This represents the best that can be achieved by any encoding/decoding of W in n dimensions.
- DeVore shows that $d_n(W_{r,q}) \geq cn^{-r/q}$ for some constant $c > 0$ depending only on q and r. So, the bound (a) is the best possible among all reasonable methods of approximating arbitrary functions in $W_{r,q}$.
- Similar considerations apply to (b).

Let us now discuss ReLU networks.

- We consider the approximation of functions $f : \mathbb{R}^q \to \mathbb{R}$ using a neural network $\mathcal{R}_{N,q}$ of functions $x \mapsto \sum_{k=1}^{N} a_k |x \cdot v_k + b_k|$.
- Consider the norm defined by

$$\|f\|_{w,q} = \operatorname{ess\,sup}_{x \in \mathbb{R}^q} \frac{|f(x)|}{\sqrt{|x|^2 + 1}}.$$

  For a function $f : \mathcal{X} \to \mathbb{R}$, the essential supremum is $\operatorname{ess\,sup} f = \inf\{a \in \mathbb{R} : \mu(f^{-1}(a, \infty)) = 0\}$.

- Denote $X_{w,q}$ the continuous functions $\mathbb{R}^q \to \mathbb{R}$ with

$$\frac{f(x)}{\sqrt{|x|^2 + 1}} \xrightarrow{|x| \to \infty} 0$$

Theorem 54 (Shallow ReLU networks)

Let $f \in W_{w,\gamma,q}$, $\gamma > 0$, where $W_{w,\gamma,q}$ is a smoothness class within $X_{w,q}$ (defined by $\|f\|_{w,\gamma,q} < \infty$ with a technical definition of the norm). Let $N \geq 1$. Then there is $P \in \mathcal{R}_{N,q}$ with

$$\|f - P\|_{w,q} < cN^{-\gamma/q}\|f\|_{w,\gamma,q}.$$

Consider a DAG $\mathcal{G}$ with a single sink (the output node). Let $\mathcal{G}X_w$ denote the $\mathcal{G}$ functions $f = \{f_v\}_{v \in V}$ with each constituent in $X_w(\mathbb{R}^{d(v)})$, and similarly $\mathcal{G}W_{w,\gamma}$.

Theorem 55 (Deep ReLU networks)

Let $1 \leq \gamma \leq 2$, $N \geq 1$, $f \in \mathcal{G}W_{w,\gamma}$, $d = \max_{v \in V} d(v)$. Then there is $P \in \mathcal{G}\mathcal{R}_N$ with

$$\|f - P\|_{\mathcal{G},w} < cN^{-\gamma/d}\|f\|_{\mathcal{G},w,\gamma}.$$

Remark 3

- If $P = \{P_v\} \in \mathcal{GR}_N$, the number of trainable parameters in each constituent network P_v is $O(N)$, and for P is $O(|V|N)$.
- When the target function is in $\mathcal{GW}_{w,\gamma}$, to obtain error at most ϵ, this translates to $O((\epsilon/V)^{-d/\gamma})$ units.
- In contrast, the shallow network, which ignores the compositional structure of the target, gives $O(\epsilon^{-q/\gamma})$ units.
- Thus, a network conforming with the function structure yields substantial savings if $d \ll q$.

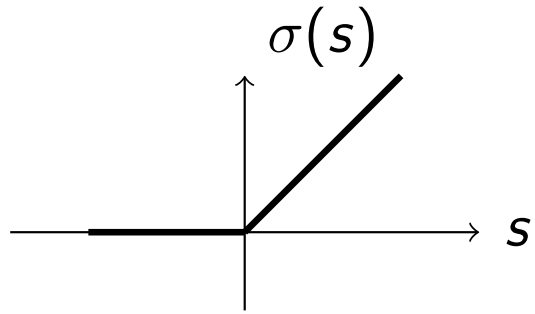
Combinatorial views on ReLUs

We follow [PMB14, MPCB14]

- Consider a Rectified Linear Unit (ReLU), which implements

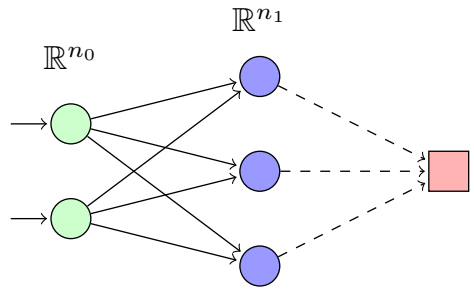
$$\sigma(s) = \max\{0, s\},$$

where s is a weighted sum of the inputs.



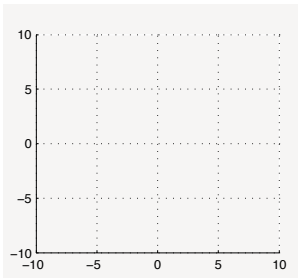
- For an n_0 dimensional input x, the jth unit in the 1st hidden layer outputs

$$h_j^1(x) = \max\{0, \sum_{i=1}^{n_0} \theta_{j,i}^{(1)} x_i + \theta_j^{(1)}\}$$

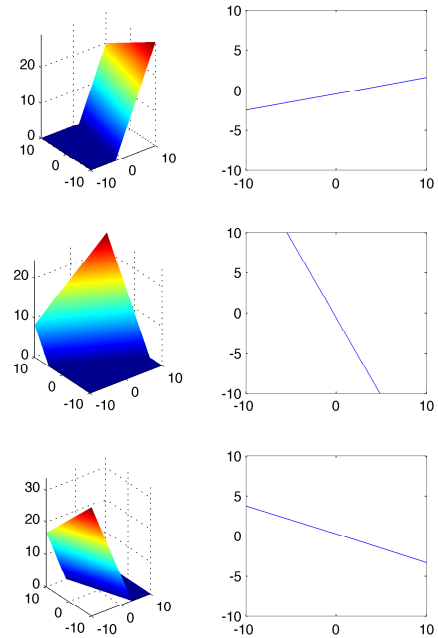


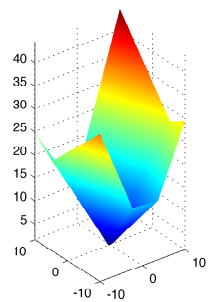
first hidden layer

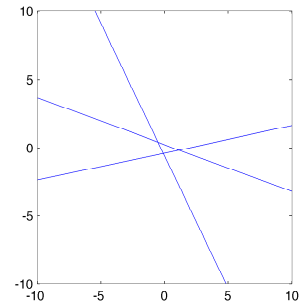
input



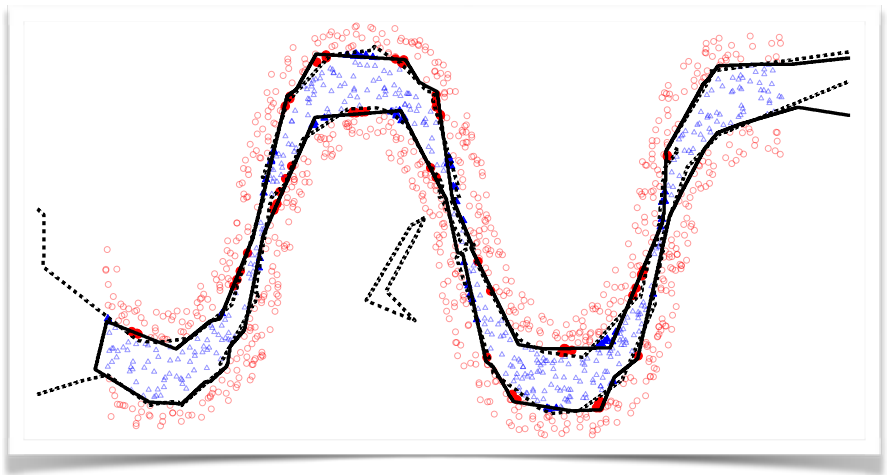
linear output







- The functions that can be represented by a ReLU network are piecewise linear.
- How many linear pieces can these functions have, depending on the structure of the network?



An exponential gap

Theorem 56 (Number of linear regions of shallow networks)

The generic (and maximum) number of linear regions of the functions computed by a feedforward network with n_0 real inputs and one layer of n_1 ReLUs is $\sum_{j=0}^{n_0} \binom{n_1}{j}$.

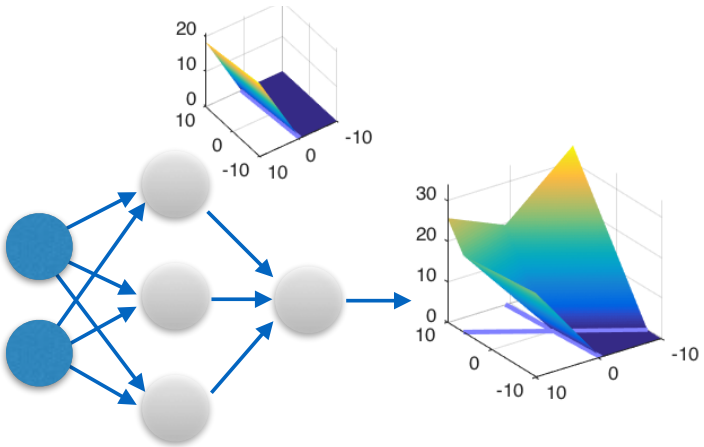
Theorem 57 (Number of linear regions of deep networks)

There are generic parameters for which the number of linear regions of the functions computed by a feedforward network with L layers of ReLUs is at least

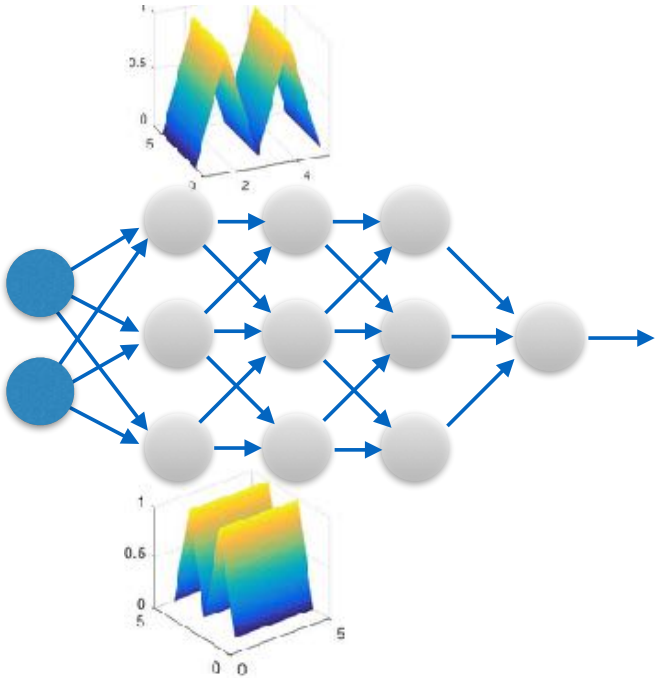
$$\prod_{l=1}^{L-1} \left\lfloor \frac{n_l}{n_0} \right\rfloor^{n_0} \cdot \sum_{j=0}^{n_0} \binom{n_L}{j}$$

In turn, representing certain piecewise linear functions requires exponentially more units and parameters when using shallow.

Proof sketch



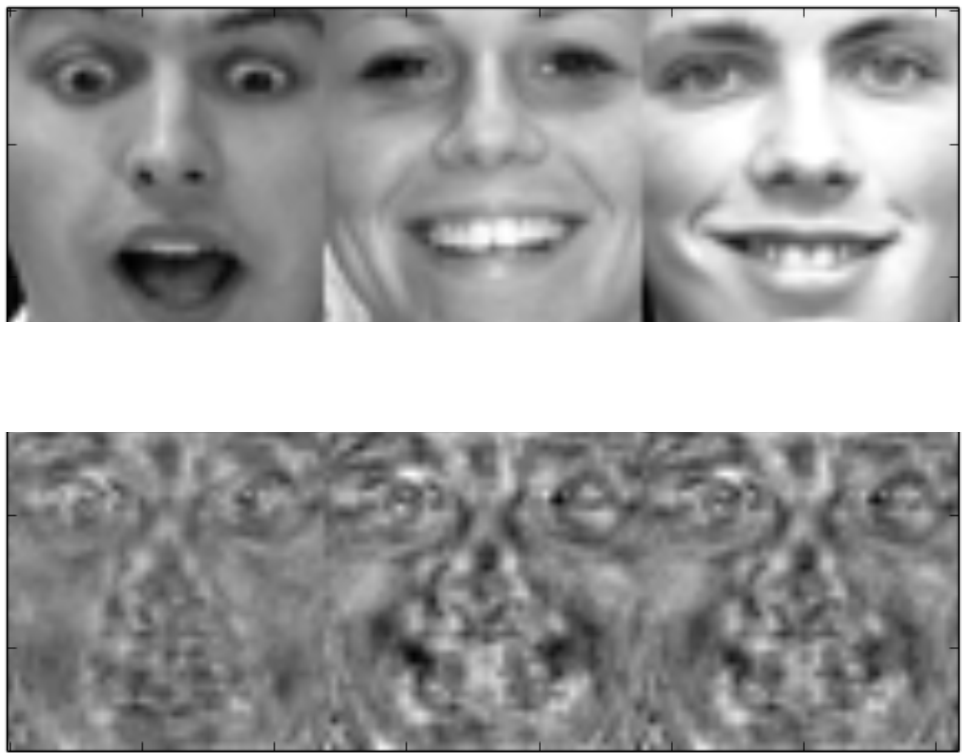
$$y_k(x) = \sum_{j=1}^{n_1} \theta_{kj}^{(2)} \sigma \left(\sum_{i=1}^{n_0} \theta_{ji}^{(1)} x_i + \theta_{j0}^{(1)} \right)$$



- Each ReLU defines a hyperplane
- The linear regions are described by a hyperplane arrangement

- Each layer folds its input space
- Last layer's arrangement replicates over folded inputs

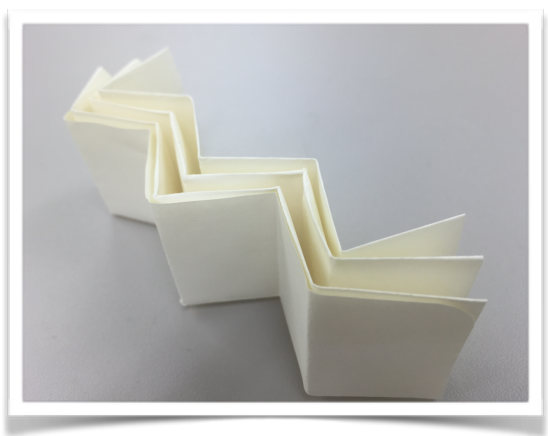
Identification of inputs

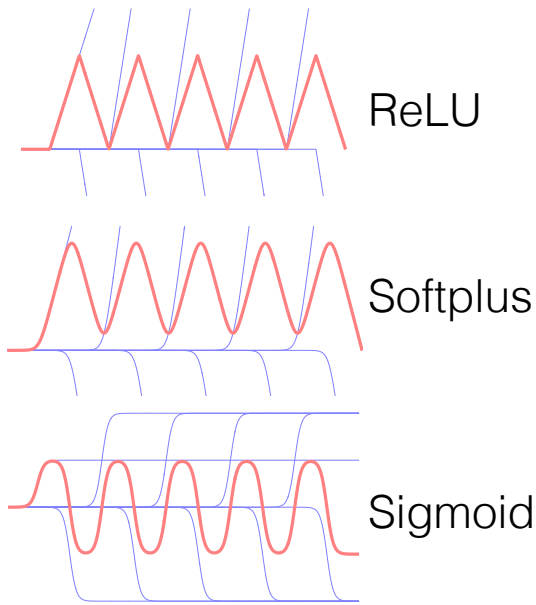


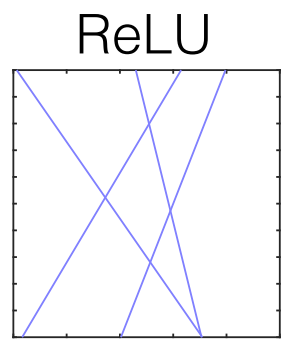
Three inputs (top) that produce the same activation of a hidden unit in a ReLU network trained in the Toronto Faces Dataset and the corresponding linear maps (bottom). [MPCB14]

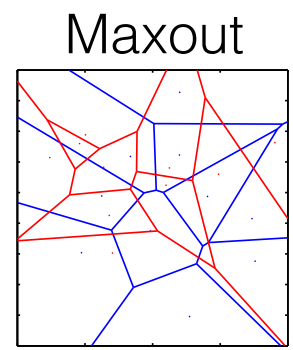
Identification of inputs

- Deep networks can replicate behaviors exponentially often
- In this way, deep networks can represent complex-looking functions with many symmetries, using only relatively few hidden units and parameters
- The notion of identification of inputs leads to exponential gaps between deep and shallow also in terms of other properties of functions and also for other types of activation functions.









Upper bounds

The number of patterns that a ReLU realizes as one traverses the input space corresponds to the number of linear regions. A hard maximum for a network with N ReLUs is 2^N. In fact

Proposition 58

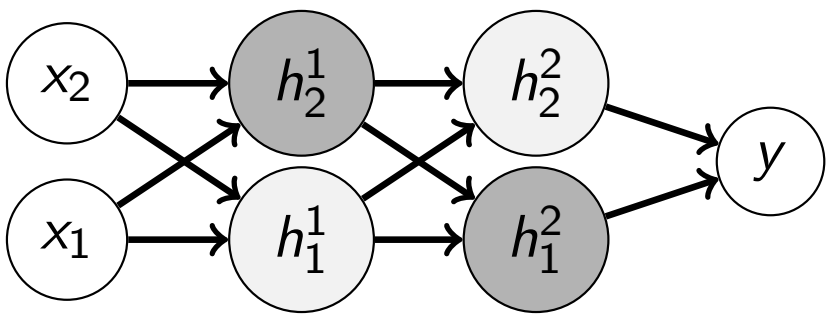
A network with n_0 inputs and layers of $n_1, \dots, n_L$ ReLUs realizes at most

$$\prod_{l=1}^{L} \sum_{j=0}^{m_{l-1}} \binom{n_l}{j}, \quad m_{l-1} = \min\{n_0, \dots, n_{l-1}\},$$

activation patterns as one traverses the input space.

Activation patterns

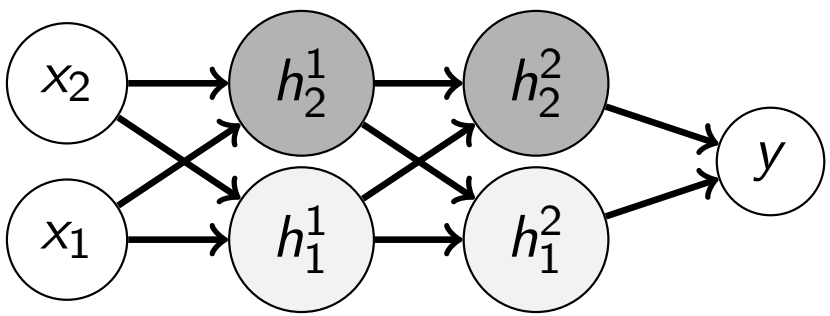
- Fix parameters $[W_l, b_l]$, $l = 1, \ldots, L$.
- At an input x we say that a unit is *active* if it outputs a value larger than 0, and else we say that it is *inactive*
- The activity pattern of all units is indicated by a vector $p = (p_1, \ldots, p_L)$, $p_l \in \{+1, -1\}^{n_l}$, $l = 1, \ldots, L$.



$p = (-, +, +, -)$

Activation patterns

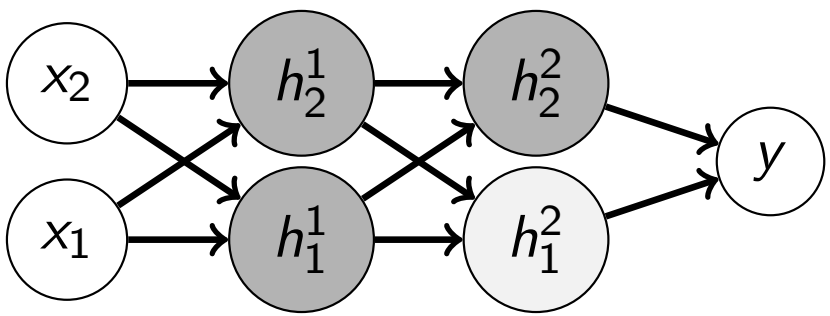
- Fix parameters $[W_l, b_l]$, $l = 1, \ldots, L$.
- At an input x we say that a unit is *active* if it outputs a value larger than 0, and else we say that it is *inactive*
- The activity pattern of all units is indicated by a vector $p = (p_1, \ldots, p_L)$, $p_l \in \{+1, -1\}^{n_l}$, $l = 1, \ldots, L$.



$p = (-, +, -, +)$

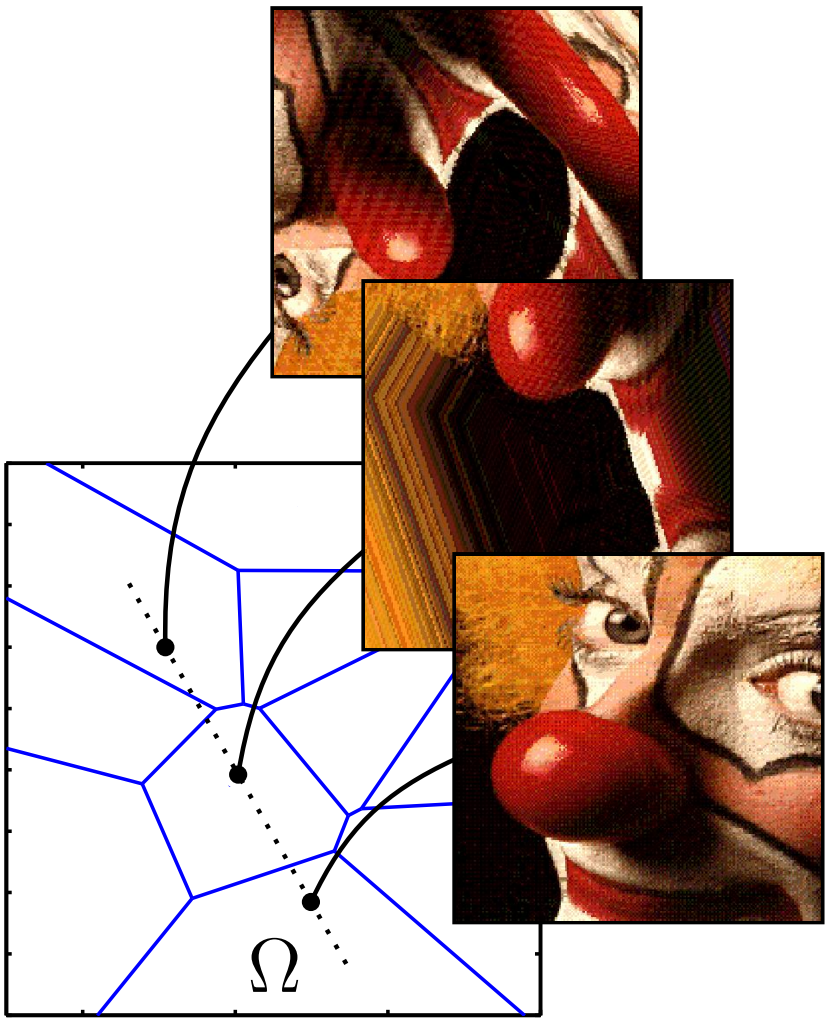
Activation patterns

- Fix parameters $[W_l, b_l]$, $l = 1, \ldots, L$.
- At an input x we say that a unit is *active* if it outputs a value larger than 0, and else we say that it is *inactive*
- The activity pattern of all units is indicated by a vector $p = (p_1, \ldots, p_L)$, $p_l \in \{+1, -1\}^{n_l}$, $l = 1, \ldots, L$.



$p = (+, +, -, +)$

Regions in parameter space



Regions in input space

Consider a network of ReLUs, representing functions

$$f(x) = \sigma(W^L \sigma(\cdots \sigma(W^1 x + b^1) \cdots) + b^L),$$

where $\sigma(s) = \max\{0, s\}$ (entrywise)

Say we want to describe

- the boundaries of the linear regions of f
- the preimage of an activation value v of the jth unit in the lth layer, $\{x : h_j^l(x) = v\}$
- the decision boundary in input space when the network has a linear threshold output unit

Regions in input space

- Write $V_p^l = \mathrm{diag}(p_l) V^l$. An input x produces the pattern p iff

$$\begin{aligned} W_p^1 x + b_p^1 &\geq 0 \\ &\vdots \\ W_p^L (\prod_{k=1}^{L-1} W_{p+}^k) x + W_p^L \sum_{k=1}^{L-1} (\prod_{m=k+1}^{L-1} W_{p+}^m) b_{p+}^k + b_p^L &\geq 0. \end{aligned} \tag{1}$$

- The input space is split into regions corresponding to the different patterns p. Some of these may be empty.
- Each region is an intersection of $n_1 + \ldots + n_L$ halfspaces. The boundary given by setting individual parts of (1) to equalities.

Regions in input space

- On input region p the network computes the linear function

$$f(x) = W^L_{p+}(\prod_{k=1}^{L-1} W^k_{p+})x + W^L_{p+} \sum_{k=1}^{L-1}(\prod_{m=k+1}^{L-1} W^m_{p+})b^k_{p+} + b^L_{p+}$$

Preimages and Decision Boundary

- For $v \in \mathbb{R}_+$, the set $\{x\colon h_i^l(x) = v\}$ is is the union of solutions, over $(p_1, \ldots, p_{l-1}) \in \{+1, -1\}^{n_1 + \ldots, + n_{l-1}}$, of the linear equation

$$W_{i:}^l (\prod_{k=1}^{l-1} W_{p+}^k) x + W_{i:}^l \sum_{k=1}^{l-1} (\prod_{m=k+1}^{l-1} W_{p+}^m) b_{p+}^k + b_i^l = v, \quad (2)$$

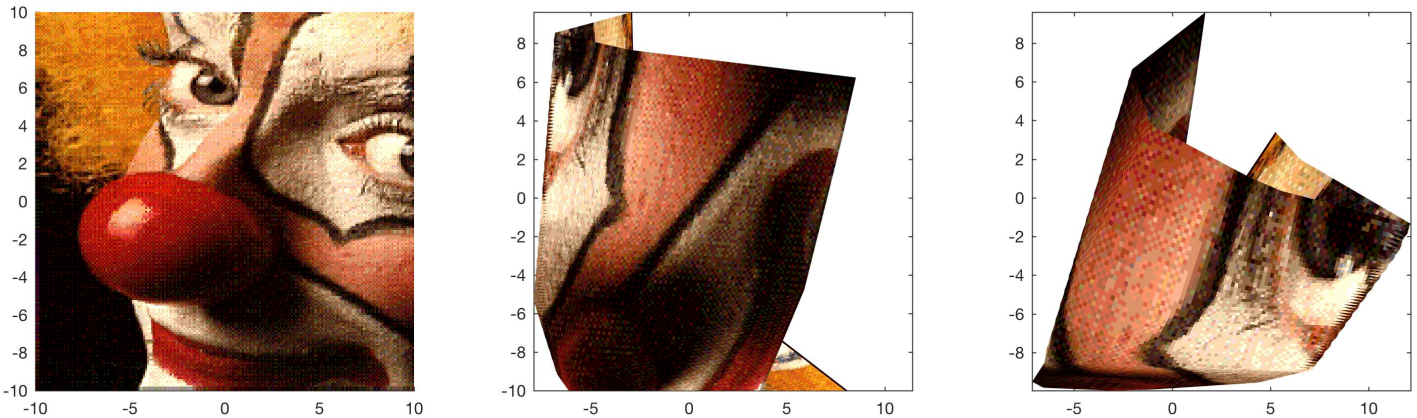
  together with the corresponding $n_1 + \ldots + n_{l-1}$ linear inequalities from (1).

- The decision boundary of the entire network would be the preimage of 0 for a linear output unit.

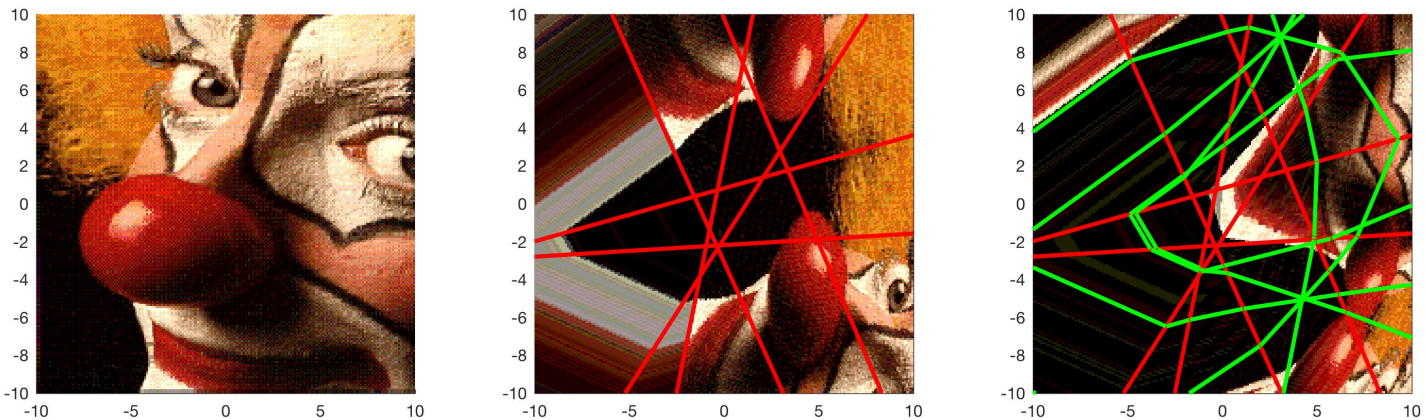
[Mon17]

Visualization

$(x, C(x)) \longrightarrow (\sigma_1(x), C(x)) \longrightarrow (\sigma_2(\sigma_1(x)), C(x))$



$(x, C(x)) \longrightarrow (x, C(\sigma_1(x))) \longrightarrow (x, C(\sigma_2(\sigma_1(x))))$



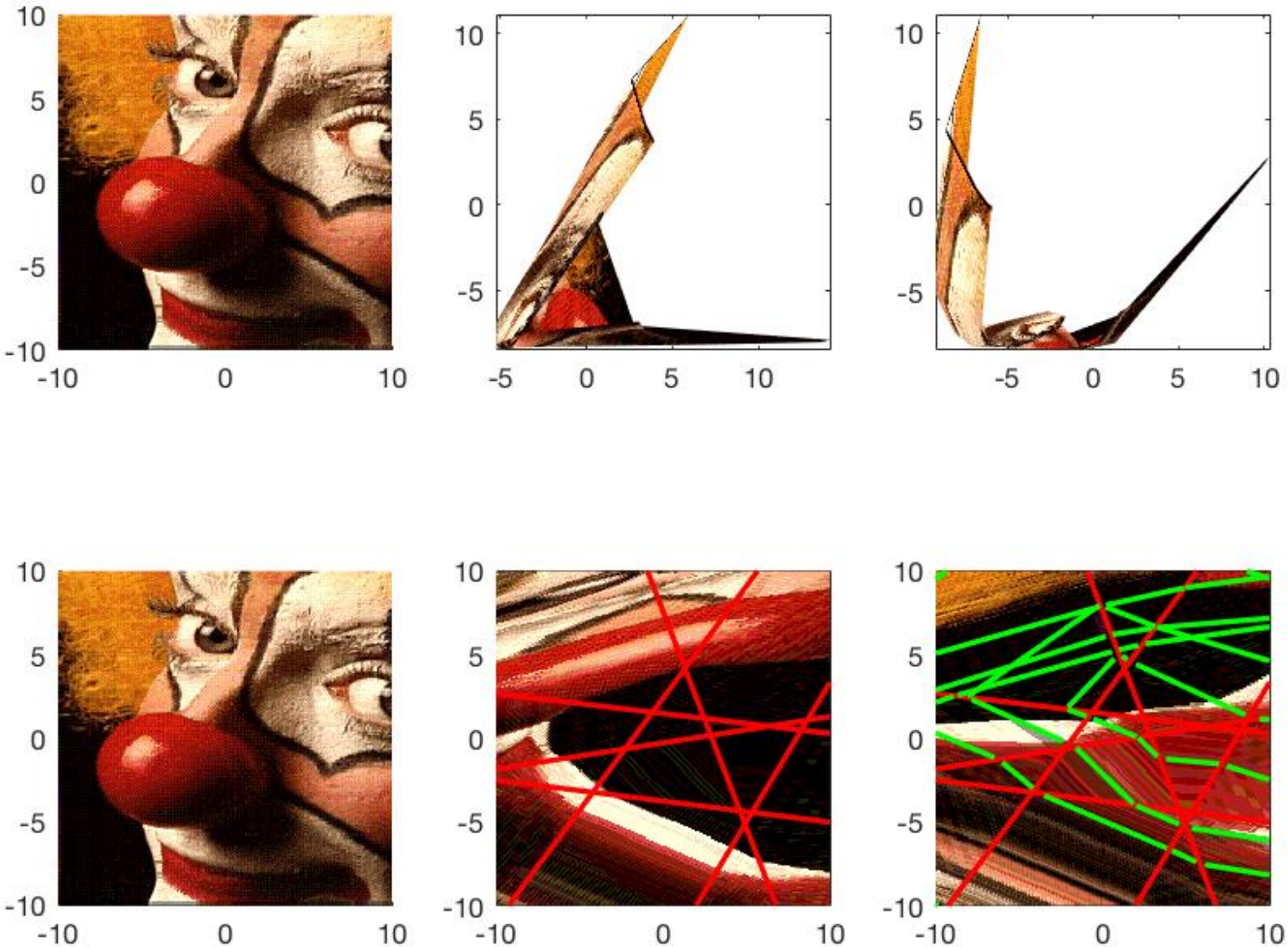
Visualization

Here $n_0 = 2$ input units, $n_1 = 6$ ReLUs followed by affine pooling with two outputs, $n_2 = 6$ ReLUs also followed by affine pooling with two outputs.

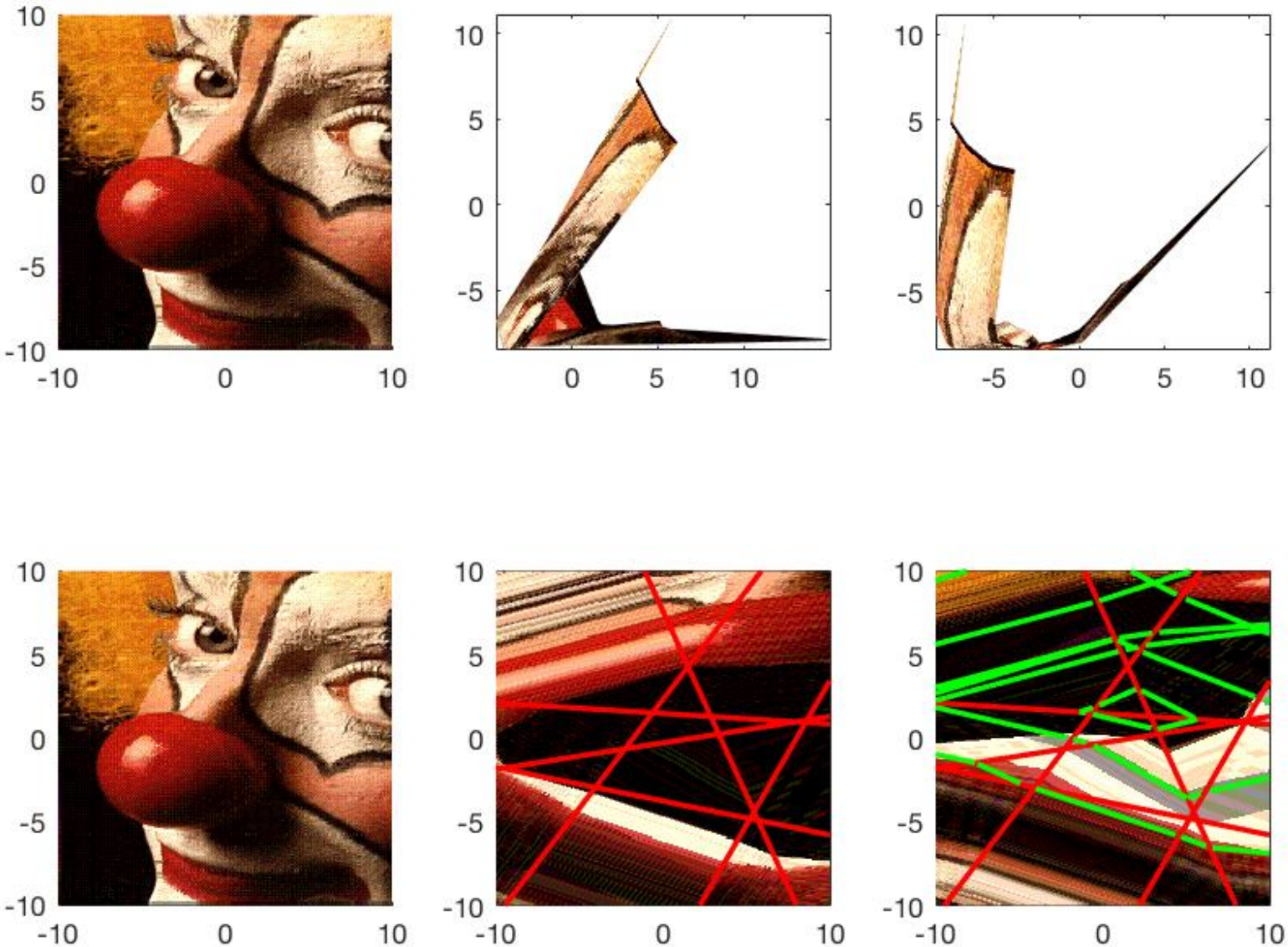
- We fix a color map $C : \mathbb{R}^2 \mapsto \texttt{Colors}$ that assigns a color to each point in the plane.
- The top row visualizes the function computed by the network by placing the input colors at the locations of the corresponding outputs.
- The bottom row visualizes the function by assigning the colors of the output values to the corresponding input values.

Red lines divide the activity regions of units at the first layer.
Green lines divide the activity regions of units at the second layer.

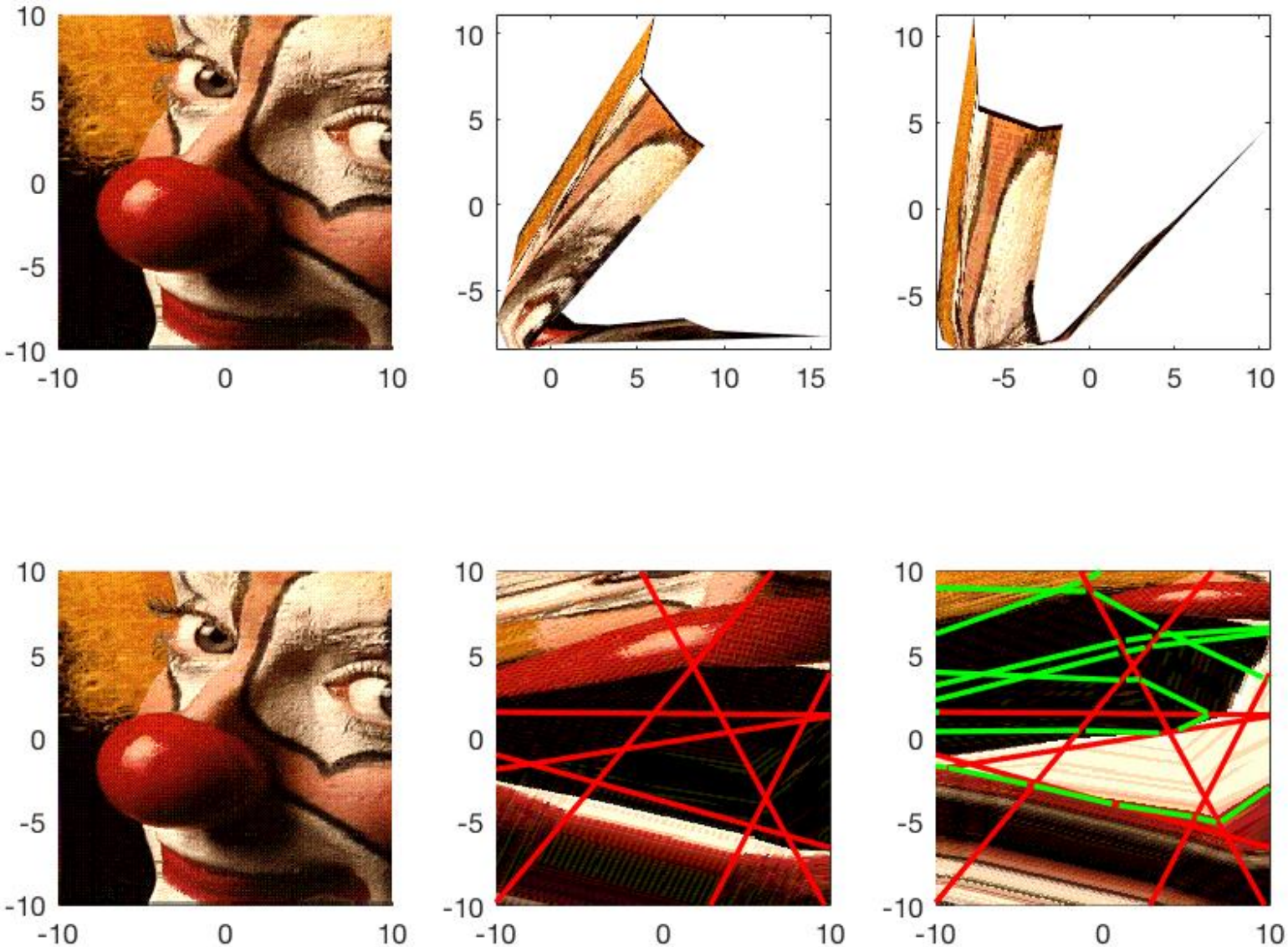
Visualization



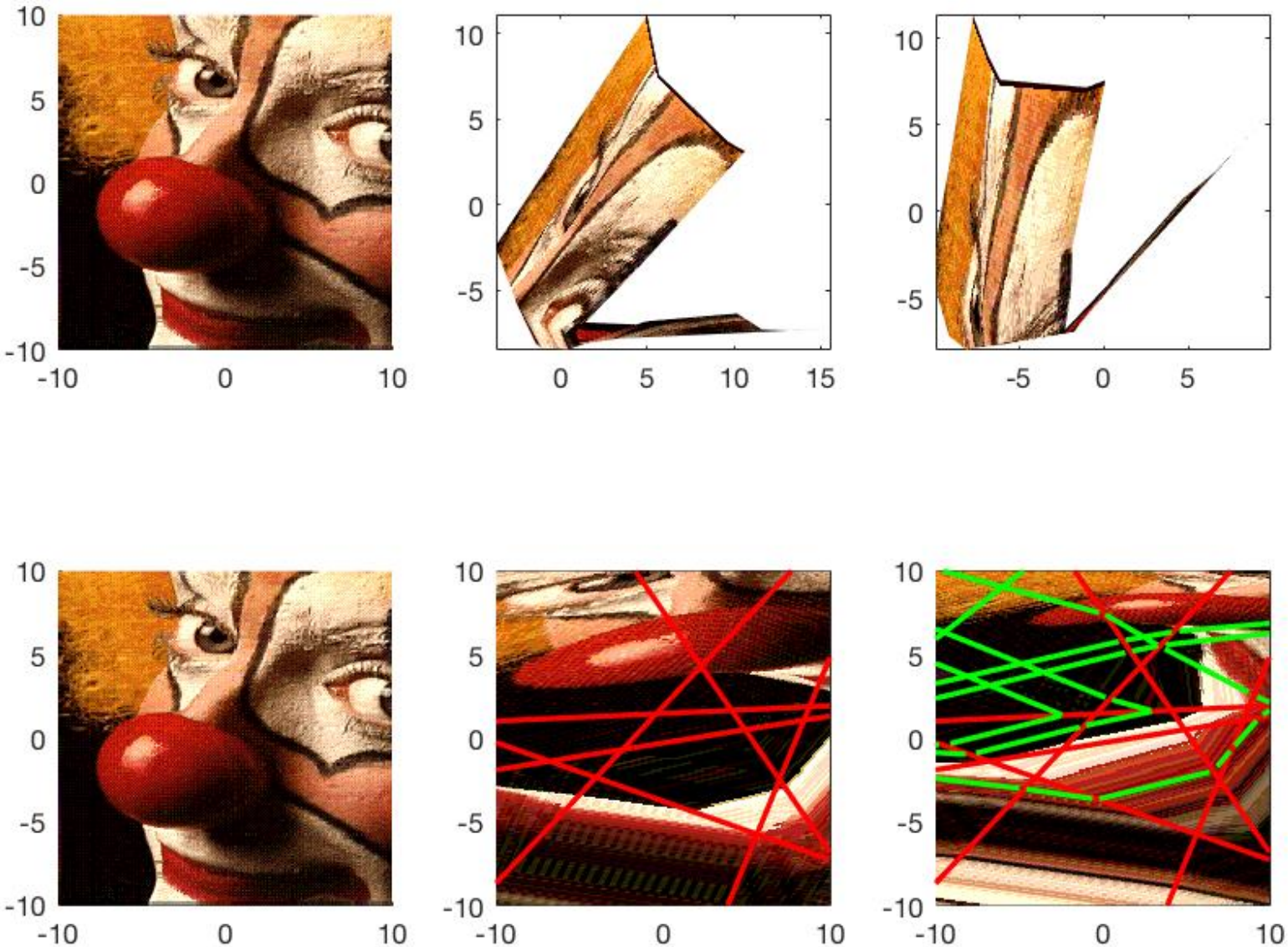
Visualization



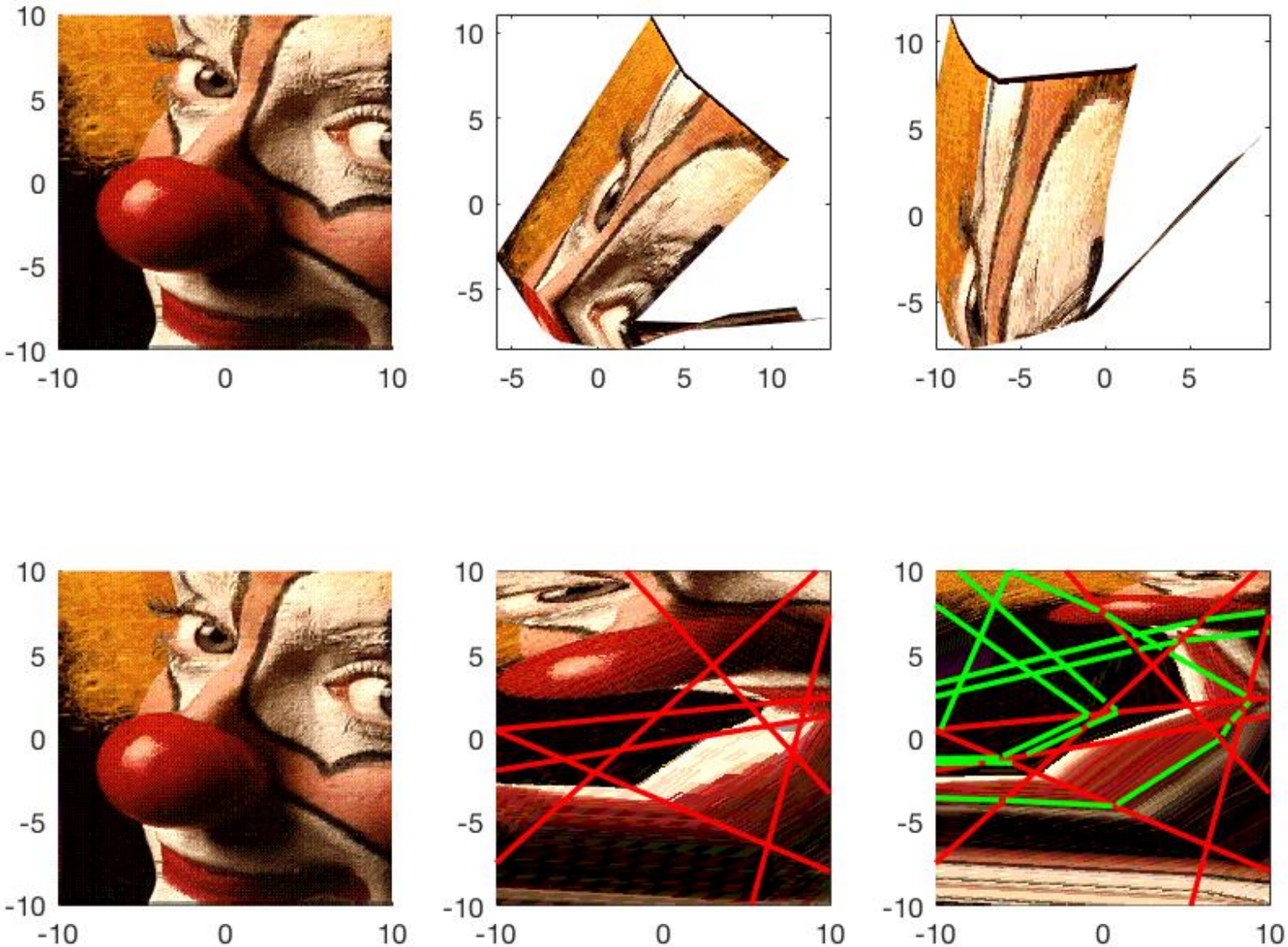
Visualization



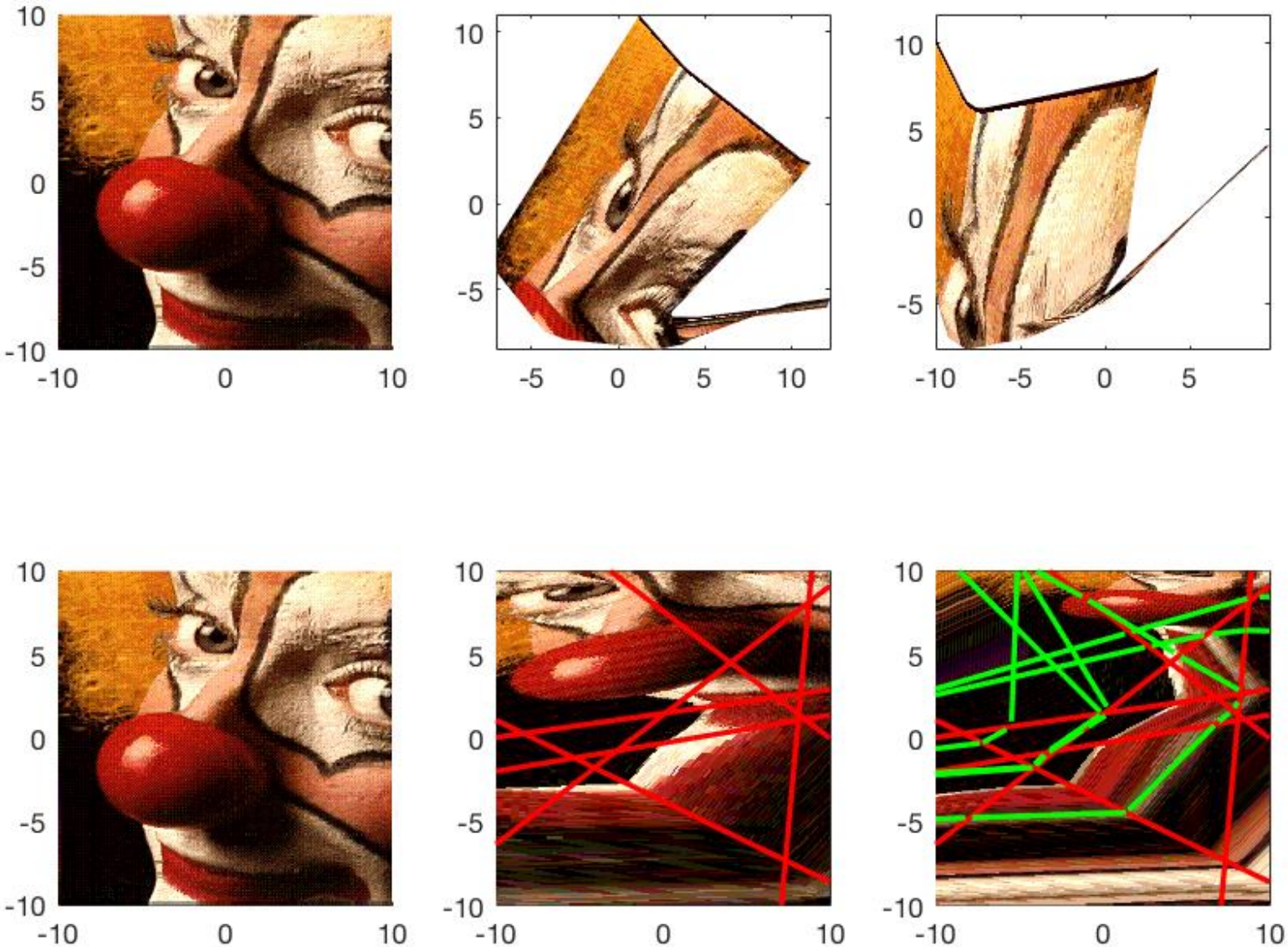
Visualization



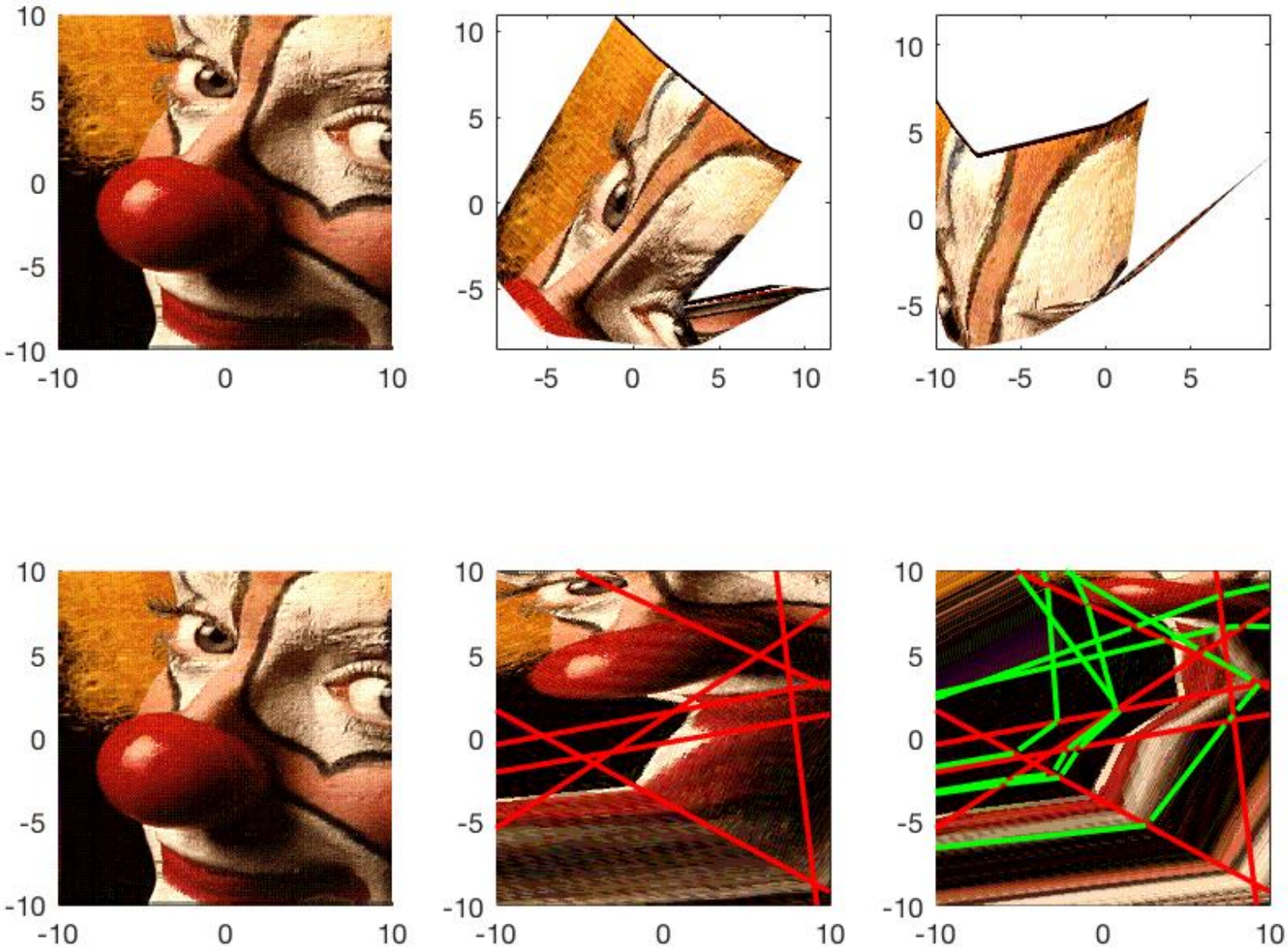
Visualization



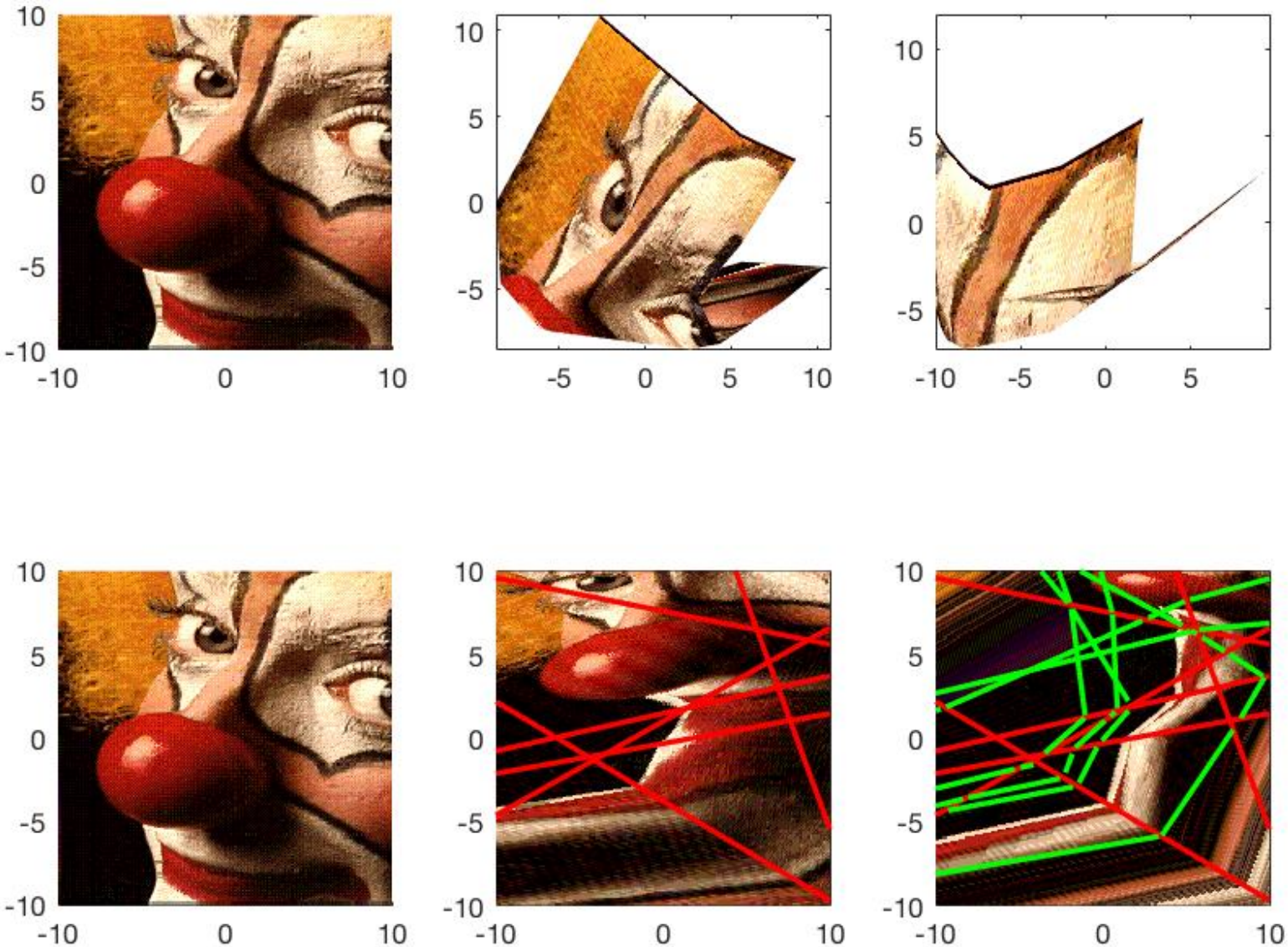
Visualization



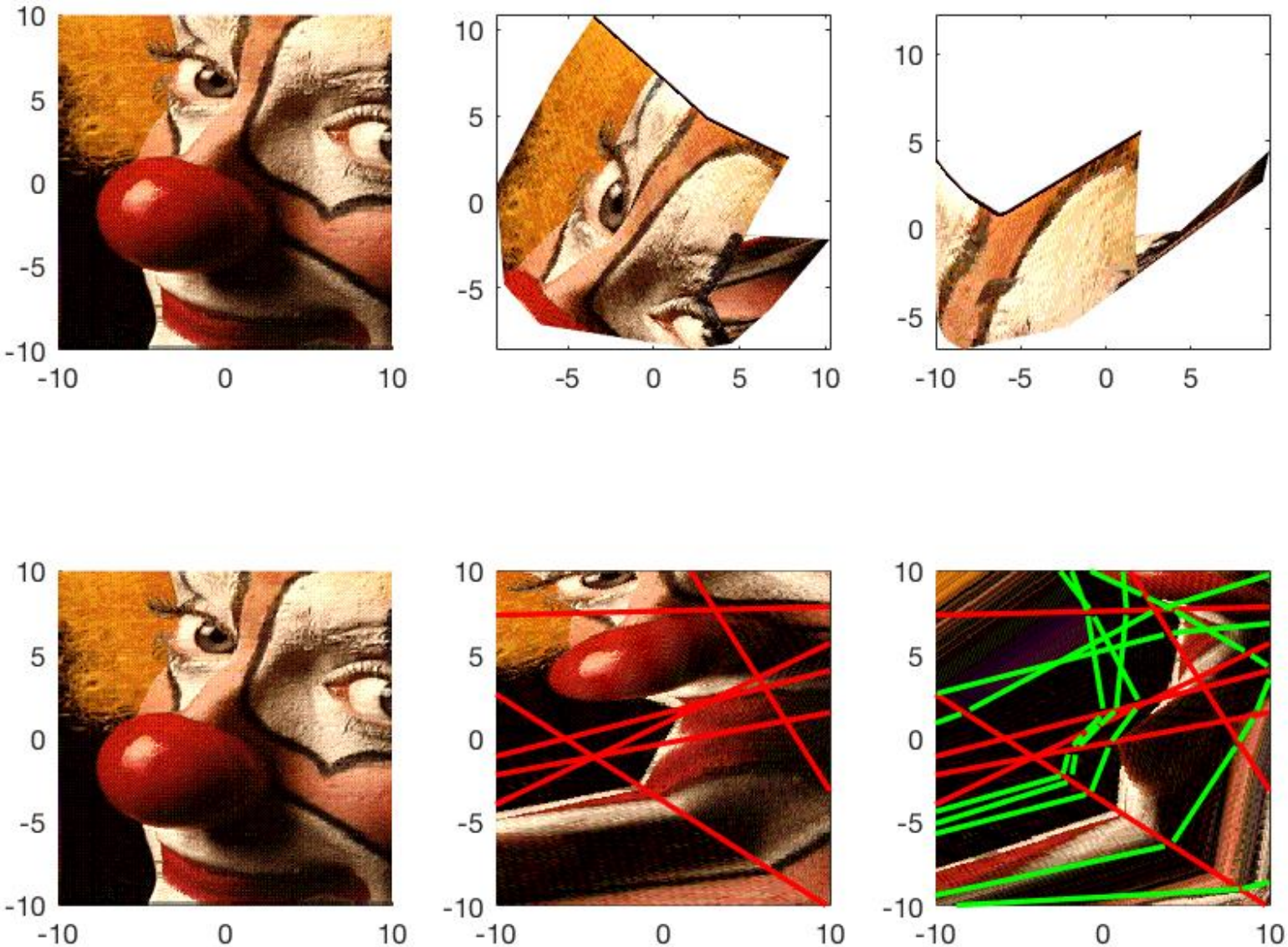
Visualization



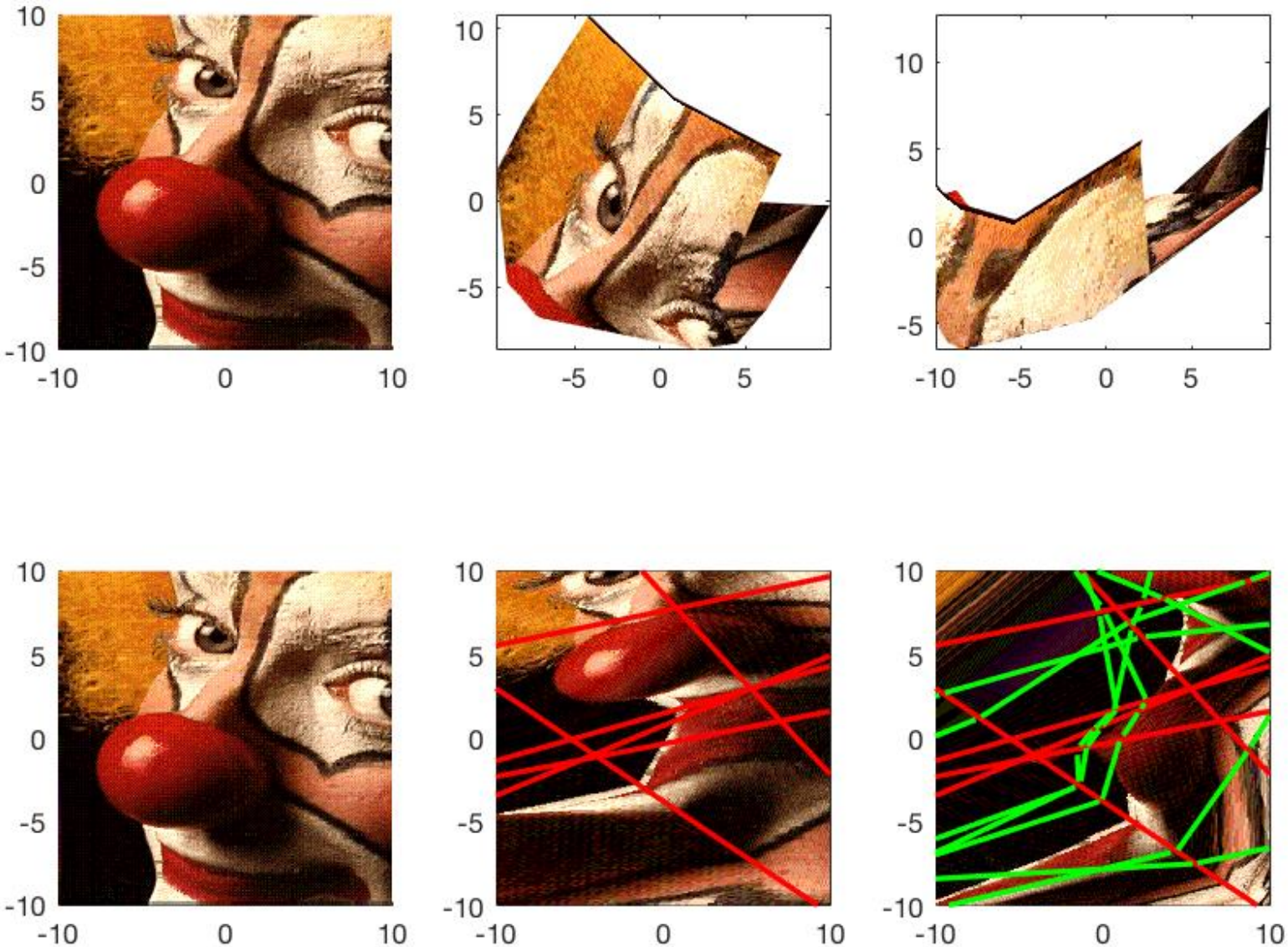
Visualization



Visualization



Visualization



Summary

1. Observations
   - The curse of dimensionality
   - Multivariate non-linear functions
   - Is learning feasible?
2. Supervised Learning Theory
   - A formal definition of learning
   - Growth function and VC dimension
   - Linear threshold networks
   - Geometric techniques
   - General bounds for neural networks
3. Approximation Properties
   - Universal approximation and rate of convergence
   - Deep and Shallow
   - Combinatorial views on ReLUs

References I

Martin Anthony and Peter L. Bartlett.

Neural Network Learning: Theoretical Foundations.

Cambridge University Press, New York, NY, USA, 1st edition, 2009.

Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin.

Learning From Data.

AMLBook, 2012.

Andrew R. Barron.

Universal approximation bounds for superpositions of a sigmoidal function.

IEEE Trans. Information Theory, 39:930–945, 1993.

Yoshua Bengio.

Learning deep architectures for AI.

Found. Trends Mach. Learn., 2(1):1–127, January 2009.

References II

Christopher M. Bishop.

Neural Networks for Pattern Recognition.

Oxford University Press, Inc., New York, NY, USA, 1995.

Johan Håstad and Mikael Goldmann.

On the power of small-depth threshold circuits.

computational complexity, 1(2):113–129, Jun 1991.

Wolfgang Maass.

Neural nets with superlinear vc-dimension.

Neural Computation, 6:877–884, 1994.

Guido Montúfar.

Notes on the number of linear regions of deep neural networks.

ResearchGate 322539221, 2017.

References III

H. N. Mhaskar and T. Poggio.

Deep vs. shallow networks: An approximation theory perspective.

Analysis and Applications, 14(06):829–848, 2016.

Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio.

On the number of linear regions of deep neural networks.

In *Advances in Neural Information Processing Systems 27*, pages 2924–2932. Curran Associates, Inc., 2014.

Razvan Pascanu, Guido Montúfar, and Yoshua Bengio.

On the number of response regions of deep feed forward networks with piece-wise linear activations.

In *International Conference on Learning Representations (ICLR 2014)*, 2014.

References IV

Brian D. Ripley and N. L. Hjort.

Pattern Recognition and Neural Networks.

Cambridge University Press, New York, NY, USA, 1st edition, 1995.

Nolan R. Wallach.

Topics in geometry.

chapter On a Theorem of Milnor and Thom, pages 331–348. Birkhauser Boston Inc., Cambridge, MA, USA, 1996.