Discrete Geometry meets Machine Learning

Amitabh Basu

Johns Hopkins University

Optimization and Discrete Geometry conference

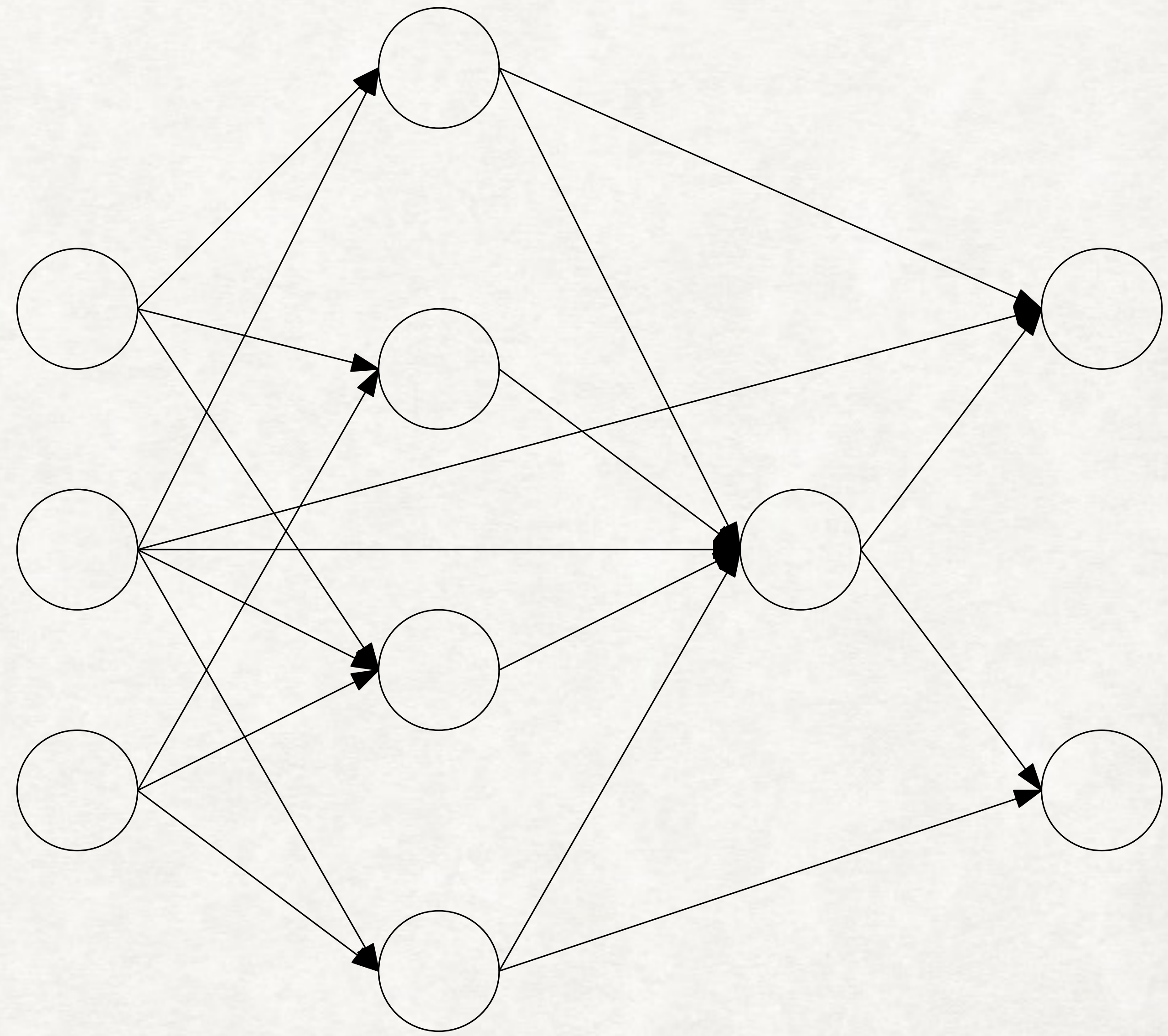
Tel-Aviv University, April 24, 2018

Joint work with Raman Arora, Poorya Mianjy, Anirbit Mukherjee

What is a Deep Neural Network (DNN) ?

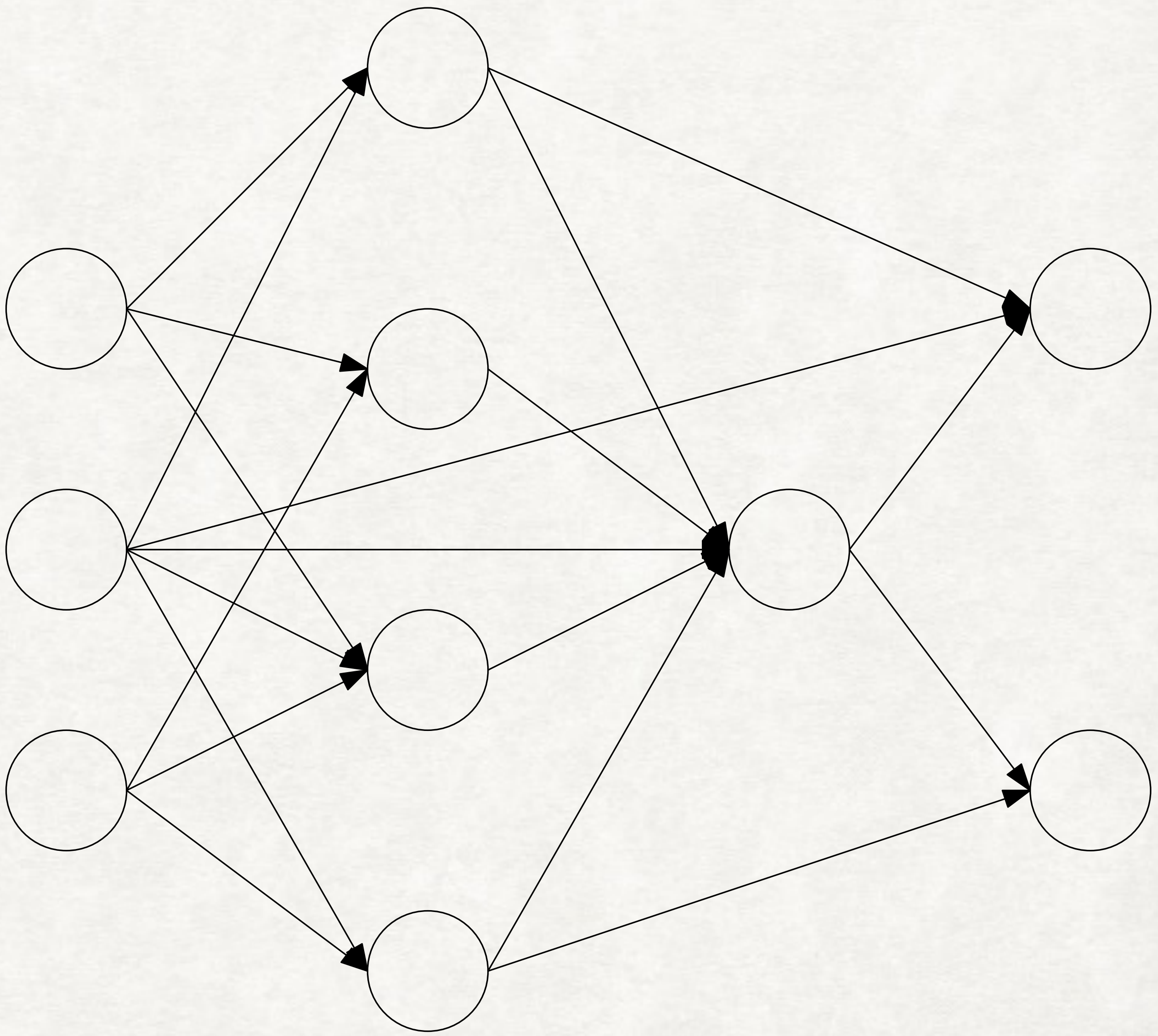
What is a Deep Neural Network (DNN) ?

- Directed Acyclic Graph (Network Architecture)



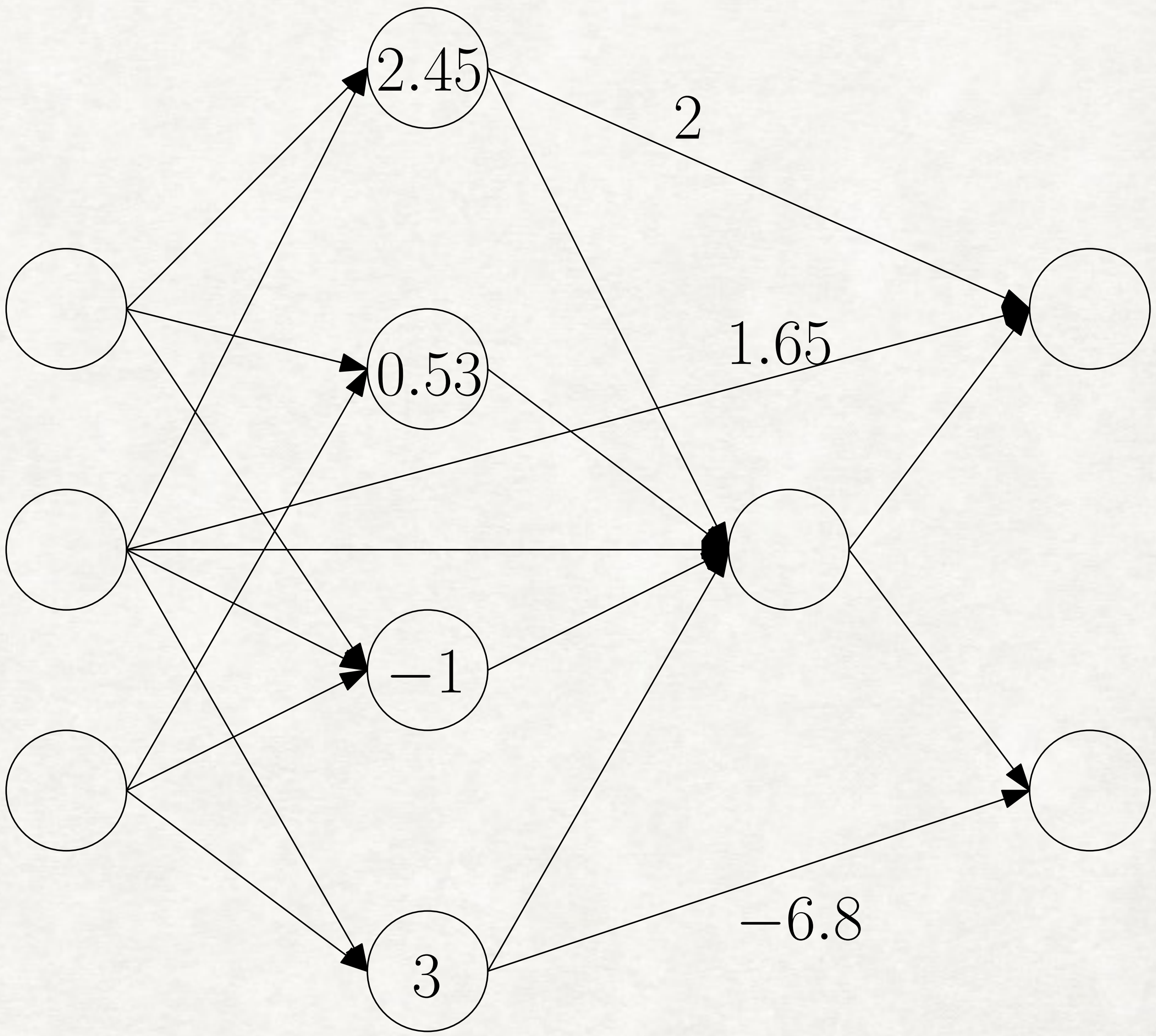
What is a Deep Neural Network (DNN) ?

- Directed Acyclic Graph (Network Architecture)
- Weights on every edge and every vertex



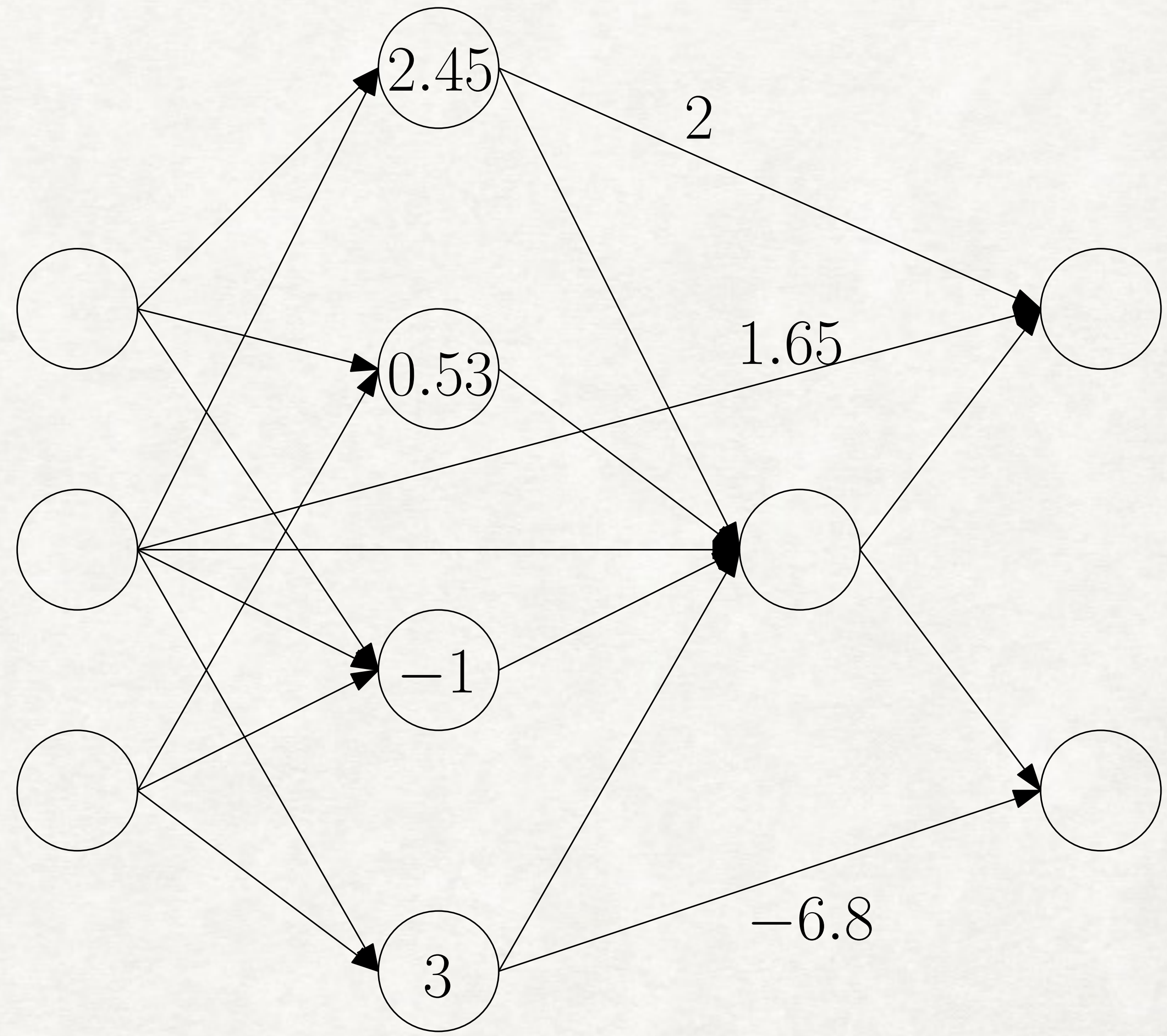
What is a Deep Neural Network (DNN) ?

- Directed Acyclic Graph (Network Architecture)
- Weights on every edge and every vertex



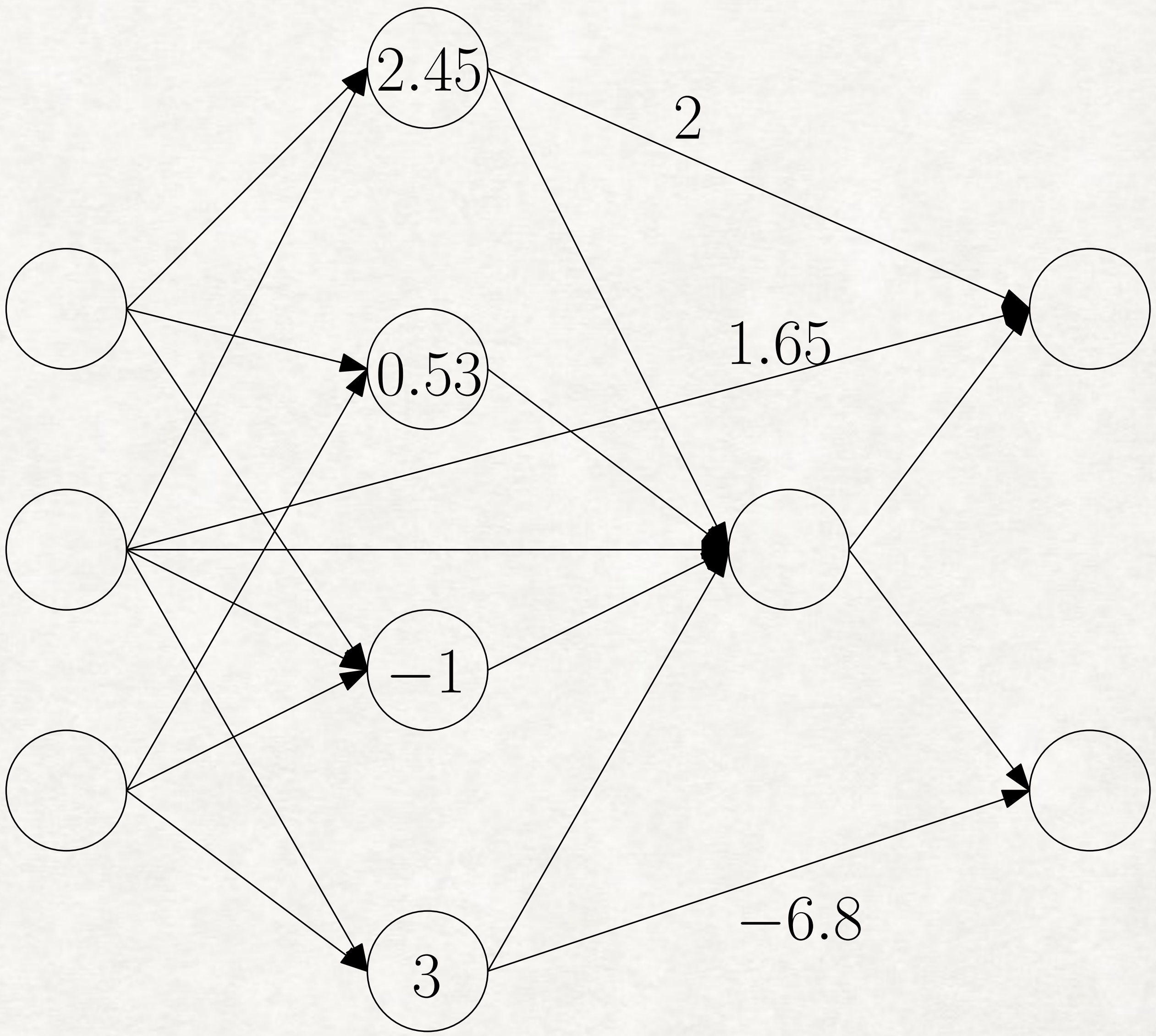
What is a Deep Neural Network (DNN) ?

- Directed Acyclic Graph (Network Architecture)
- Weights on every edge and every vertex
- R -> R "Activation Function"
  Examples:
  f(x) = max{0,x} – Rectified Linear Unit (ReLU)
  f(x) = eˣ/(1 + eˣ) – Sigmoid



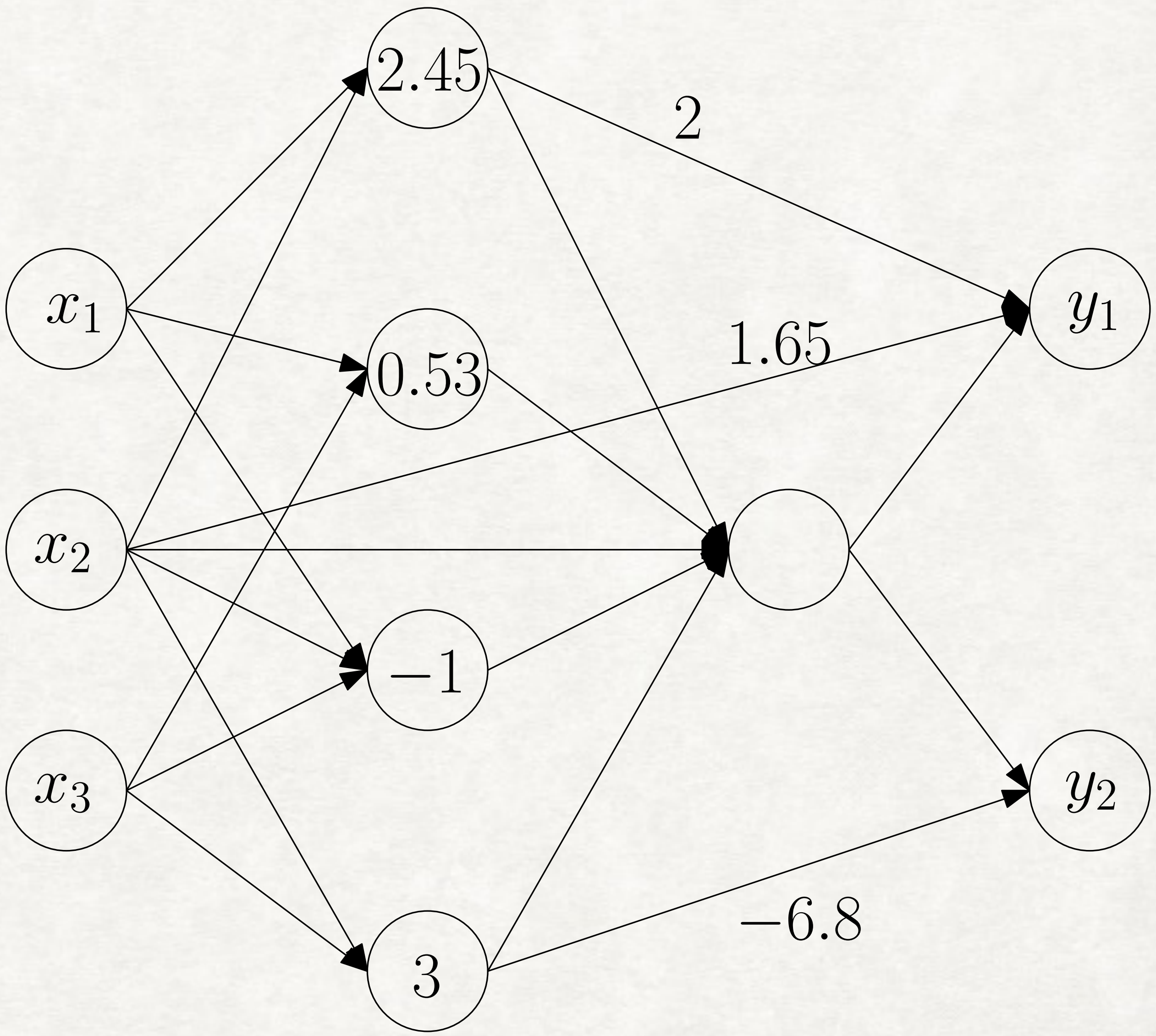
What is a Deep Neural Network (DNN) ?

- Directed Acyclic Graph (Network Architecture)
- Weights on every edge and every vertex
- R -> R "Activation Function"
  Examples:
  $f(x) = \max\{0,x\}$ – Rectified Linear Unit (ReLU)
  $f(x) = e^x/(1 + e^x)$ – Sigmoid
- Sources = input, Sinks = output



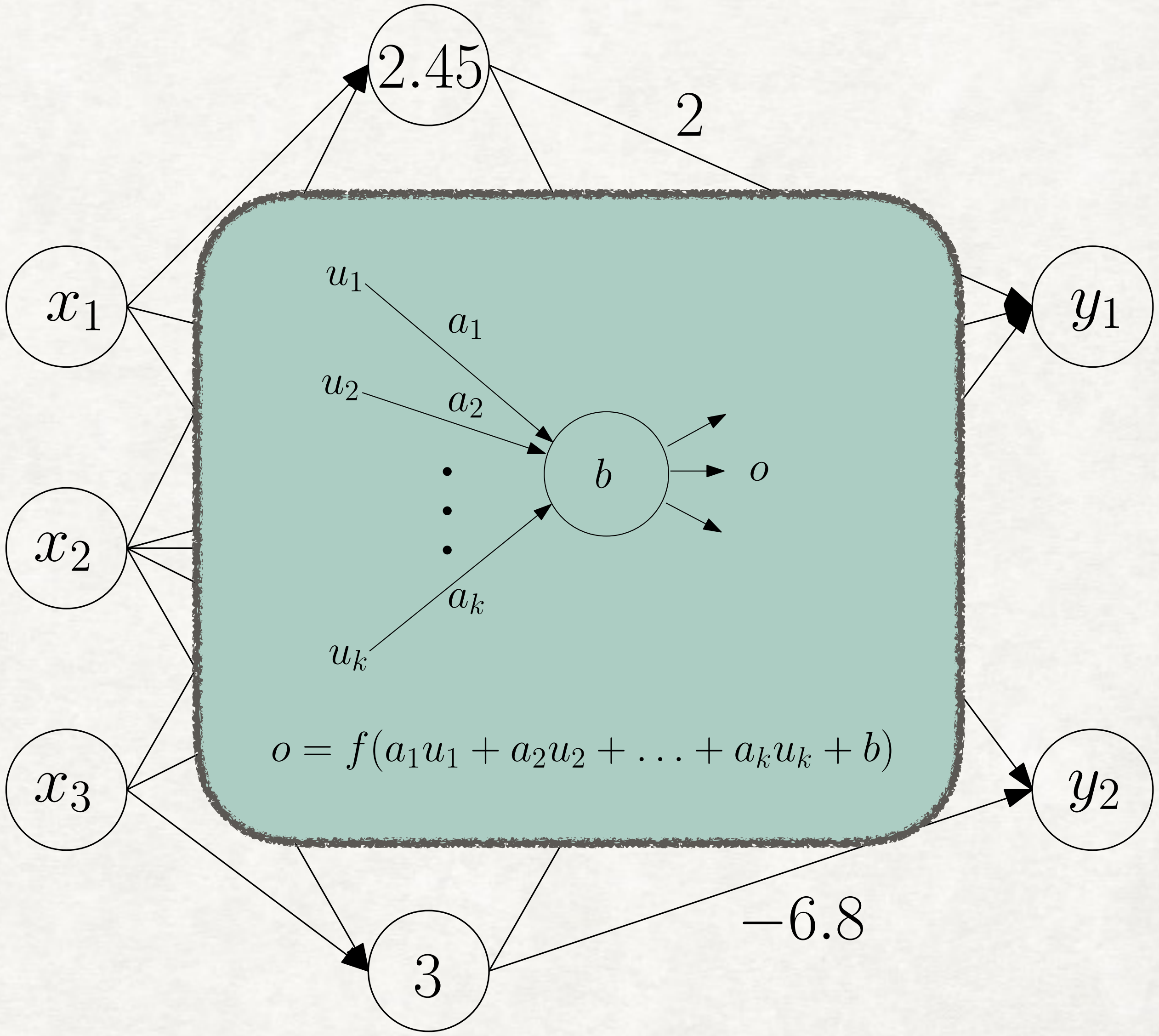
What is a Deep Neural Network (DNN) ?

- Directed Acyclic Graph (Network Architecture)
- Weights on every edge and every vertex
- R -> R "Activation Function"
  Examples:
  f(x) = max{0,x} – Rectified Linear Unit (ReLU)
  f(x) = e^x/(1 + e^x) – Sigmoid
- Sources = input, Sinks = output



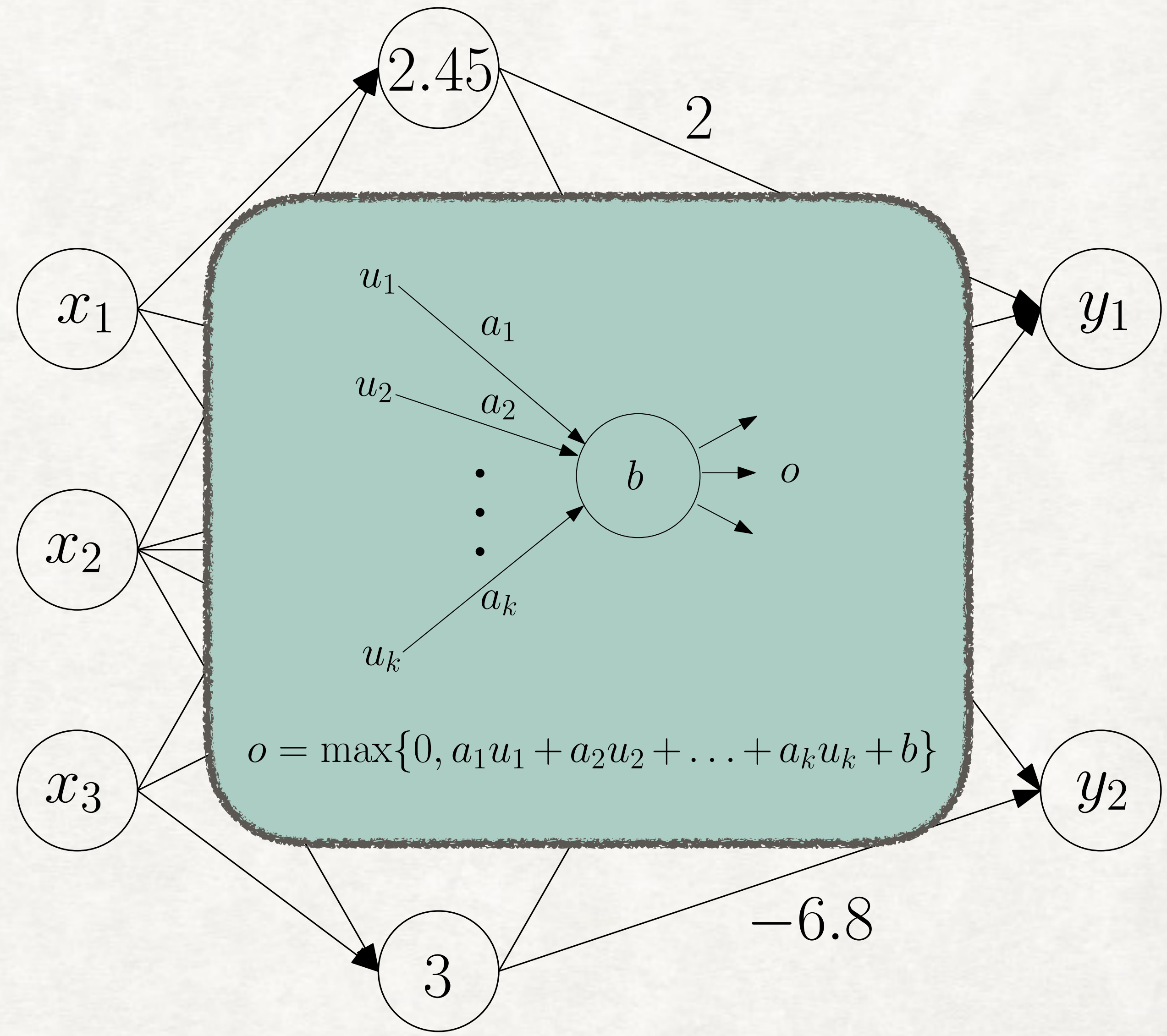
What is a Deep Neural Network (DNN) ?

- Directed Acyclic Graph (Network Architecture)
- Weights on every edge and every vertex
- R -> R "Activation Function"
  Examples:
  f(x) = max{0,x} – Rectified Linear Unit (ReLU)
  f(x) = $e^x/(1 + e^x)$ – Sigmoid
- Sources = input, Sinks = output



What is a Deep Neural Network (DNN) ?

- Directed Acyclic Graph (Network Architecture)
- Weights on every edge and every vertex
- R -> R "Activation Function"
  Examples:
  f(x) = max{0,x} – Rectified Linear Unit (ReLU)
  f(x) = e^x/(1 + e^x) – Sigmoid
- Sources = input, Sinks = output



Problems of interest for DNNs

- Expressiveness: What family of functions can one represent using DNNs?

- Efficiency: How many layers (depth) and vertices (size) needed represent functions in the family?

- Training the network: Given architecture, data points (x,y), find weights for the "best fit" function.

- Generalization error: Rademacher complexity, VC dimension

Problems of interest for DNNs

- Expressiveness: What family of functions can one represent using DNNs?

- Efficiency: How many layers (depth) and vertices (size) needed represent functions in the family?

- Training the network: Given architecture, data points (x,y), find weights for the "best fit" function.

- Generalization error: Rademacher complexity, VC dimension

Calculus of DNN functions

Calculus of DNN functions

- f_1 in DNN(k_1,s_1), f_2 in DNN(k_2,s_2) => $f_1 + f_2$ in DNN($\max\{k_1,k_2\}$, s_1+s_2)
- f in DNN(k,s), c in R => cf in DNN(k,s)
- f_1 in DNN(k_1,s_1), f_2 in DNN(k_2,s_2) => $f_1 \circ f_2$ in DNN(k_1+k_2, s_1+s_2)
- f_1 in ReLU-DNN(k_1,s_1), f_2 in ReLU-DNN(k_2,s_2) =>
  $\max\{f_1, f_2\}$ in ReLU-DNN($\max\{k_1,k_2\}+1$, s_1+s_2+4)
- Affine functions can be implemented in ReLU-DNN(1,2n)

Problems of interest for DNNs

- Expressiveness: What family of functions can one represent using DNNs?

- Efficiency: How many layers (depth) and vertices (size) needed to represent functions in the family?

- Training the network: Given architecture, data points (x,y), find weights for the "best fit" function.

- Generalization error: Rademacher complexity, VC dimension

Expressiveness of ReLU DNNs

Theorem (Arora, Basu, Mianjy, Mukherjee 2016): Any ReLU DNN with 'n' inputs implements a continuous piecewise affine function on R^n. Conversely, any continuous piecewise affine function on R^n can be implemented by some ReLU DNN. Moreover, at most log(n+1) hidden layers are needed.

Expressiveness of ReLU DNNs

Theorem (Arora, Basu, Mianjy, Mukherjee 2016): Any ReLU DNN with 'n' inputs implements a continuous piecewise affine function on R^n. Conversely, any continuous piecewise affine function on R^n can be implemented by some ReLU DNN. Moreover, at most log(n+1) hidden layers are needed.

Proof: Result from circuits literature [Wang and Sun 2006] says any continuous piecewise affine function can be written as

$$c_1\max\{l^1{}_1, l^1{}_2, \ldots, l^1{}_{n+1}\} + \ldots + c_k\max\{l^k{}_1, l^k{}_2, \ldots, l^k{}_{n+1}\}$$

Expressiveness of ReLU DNNs

Theorem (Aro … **16):** Any ReLU DNN with 'n' … ous piecewise affine functi … y continuous piecewise affi … mplemented by some ReLU DN … hidden layers are needed.

Proof: Result … and Sun 2006] says any con … ction can be written as

$$c_1 \max\{l^1{}_1, l^1{}_2, \ldots, l^1{}_{n+1}\} + \ldots + c_k \max\{l^k{}_1, l^k{}_2, \ldots, l^k{}_{n+1}\}$$

Open Question

ReLU-DNN(1, *)

$\subsetneq$

ReLU-DNN(2, *)

$\subsetneq$

ReLU-DNN(3, *)

$\subsetneq$

ReLU-DNN(log(n+1), *)

Problems of interest for DNNs

- Expressiveness: What family of functions can one represent using DNNs?
- Efficiency: How many layers (depth) and vertices (size) needed to represent functions in the family?
- Training the network: Given architecture, data points (x,y), find weights for the "best fit" function.
- Generalization error: Rademacher complexity, VC dimension

Depth v/s size tradeoffs for ReLU DNNs

Theorem (Arora, Basu, Mianjy, Mukherjee 2016): For every natural number N, there exists a family of R -> R functions such that for any function f in this family, we have:

1. f is in ReLU-DNN(N^2, N^3).
2. f is NOT in ReLU-DNN(N, $(1/2)N^N - 1$).

Moreover, this family is in one-to-one correspondence with the torus in N dimensions.

Remark: More general versions, Approximation versions. n>=2 version using zonotopal norms.

Depth v/s size tradeoffs for ReLU DNNs

Fact: Any R -> R PWL function with p pieces is in ReLU-DNN(1,p+1)

Theor… every
natur… …ctions
such t…

1. f is in ReLU-DNN(N^2, N^3).
2. f is NOT in ReLU-DNN(N, (1/2)N^N - 1).

Moreover, this family is in one-to-one correspondence with the torus in N dimensions.

Remark: More general versions, Approximation versions. n>=2 version using zonotopal norms.

Depth v/s size tradeoffs for ReLU DNNs

Fact: Any R -> R PWL function with p pieces is in ReLU-DNN(1,p+1)

Theor ... every

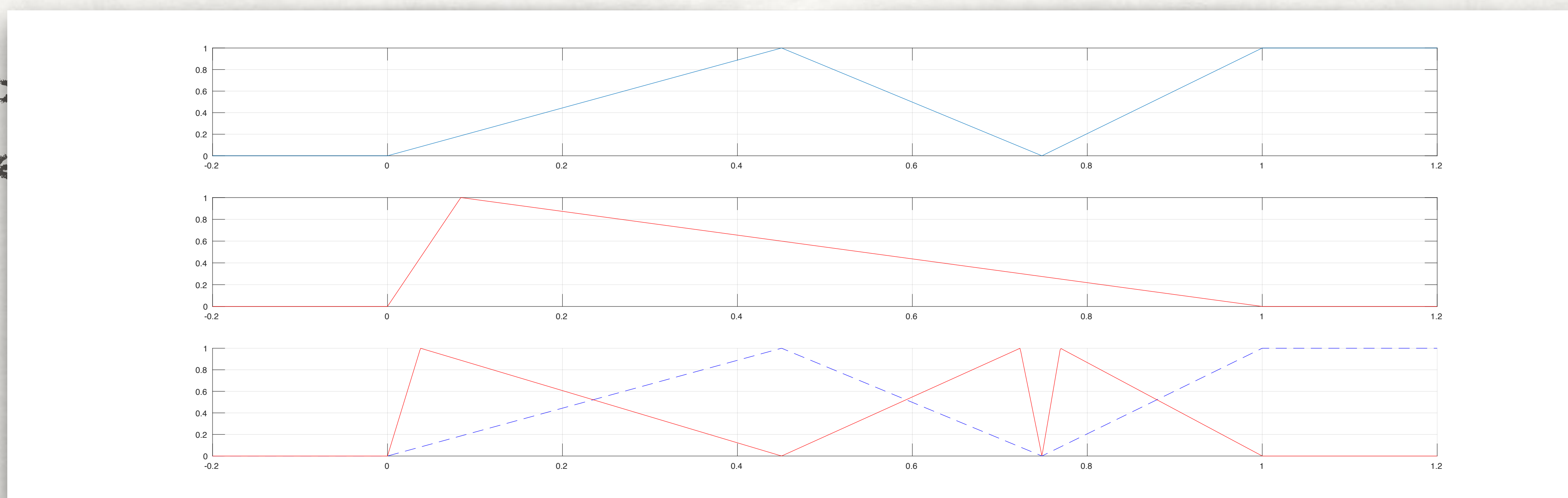
natur ... ctions

such t

1. f is in ReLU-DNN(N^2, N^3).
2. f is NOT in ReLU-DNN(N, (1/2)N^N – 1).

Mo ... ith

the

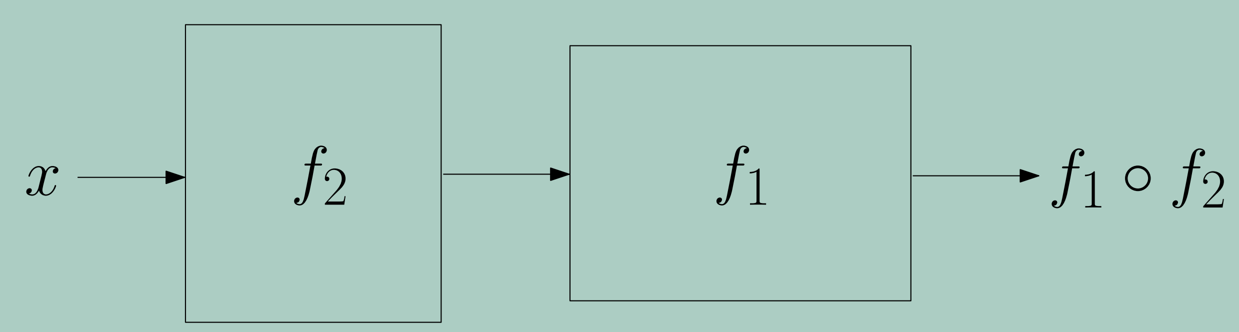


Calculus of DNN functions

- f_1 in DNN(k_1,s_1), f_2 in DNN(k_2,s_2) => $f_1 + f_2$ in DNN(max{k_1,k_2}, s_1+s_2)

- f in DNN(k,s),

- f_1 in DNN(k_1,s



(k_1+k_2, s_1+s_2)

Depth v/s size tradeoffs for ReLU DNNs

Fact: Any R -> R PWL function with p pieces is in ReLU-DNN(1,p+1)

Theor every

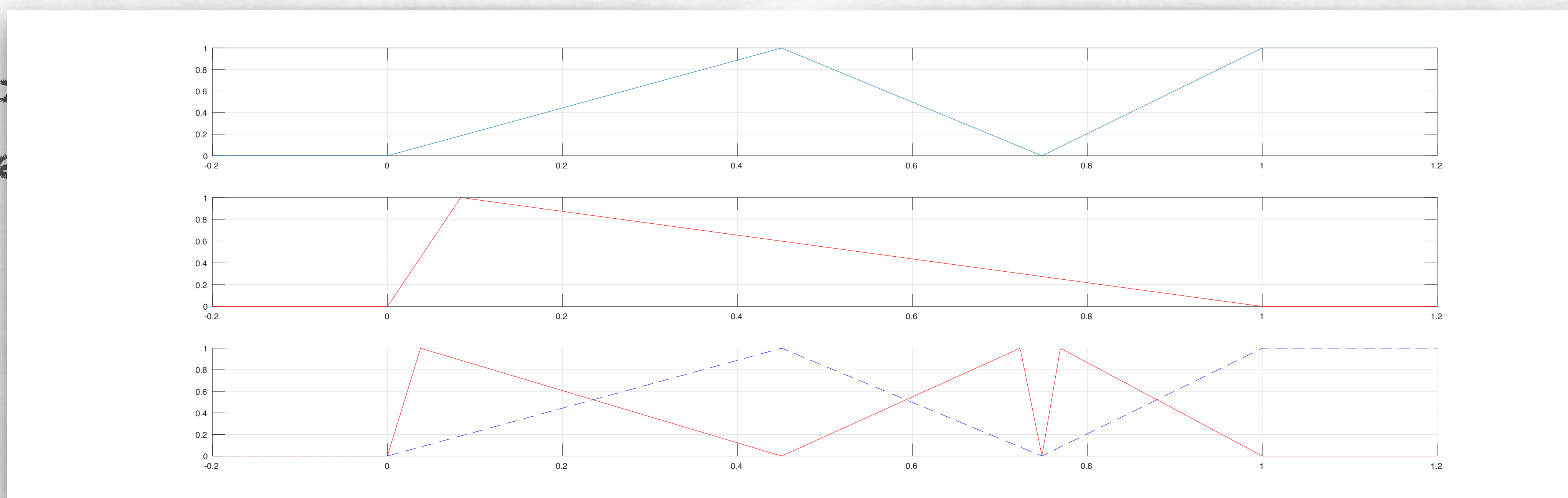
natur ctions

such t

1. f is in ReLU-DNN(N^2, N^3).
2. f is NOT in ReLU-DNN(N, $(1/2)N^N - 1$).

Mo ith

the



Depth v/s size tradeoffs for ReLU DNNs

Fact: Any R -> R function in ReLU(k, w) has at most O(w^k) pieces

Theor ... every natur ... ctions such t

1. f is in ReLU-DNN(N^2, N^3).
2. f is NOT in ReLU-DNN(N, $(1/2)N^N - 1$).

Moreover, this family is in one-to-one correspondence with the torus in N dimensions.

Remark: More general versions, Approximation versions. n>=2 version using zonotopal norms.

Depth v/s size tradeoffs for ReLU DNNs

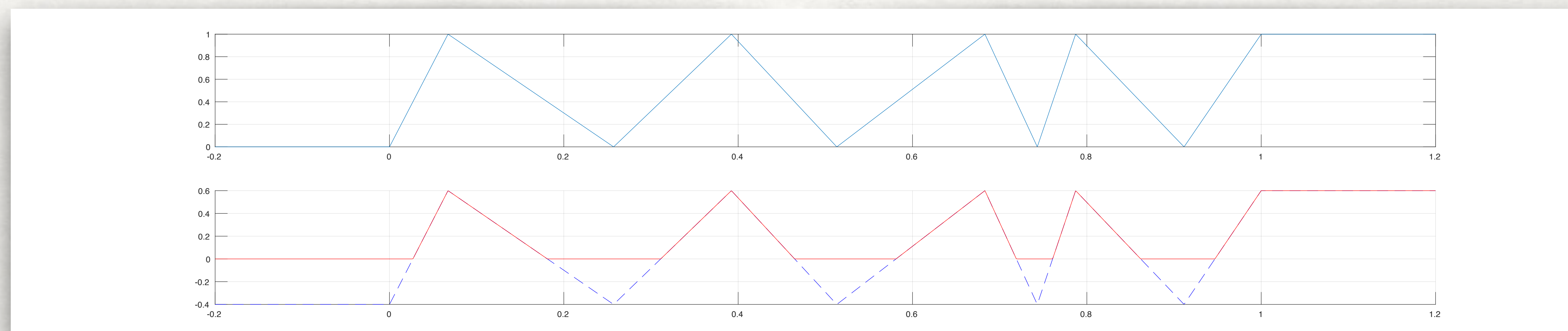
Theor ... every

natur ... ctions

such t

Fact: Any R -> R function in ReLU(k, w) has at most O(w^k) pieces

1. f is in ReLU-DNN(N^2, N^3).
2. f is NOT in ReLU-DNN(N, (1/2)N^N – 1).



n>=2 version using zonotopal norms.

Depth v/s size tradeoffs for ReLU DNNs

Theorem (Arora, Basu, Mianjy, Mukherjee 2016): For every natural number N, there exists a family of R -> R functions such that for any function f in this family, we have:

1. f is in ReLU-DNN(N^2, N^3).
2. f is NOT in ReLU-DNN(N, (1/2)N^N - 1).

Moreover, this family is in one-to-one correspondence with the torus in N dimensions.

Remark: More general versions, Approximation versions. n>=2 version using zonotopal norms.

Depth v/s size tradeoffs for ReLU DNNs

Open Questions

1. Finer gaps and n >= 2. Recent result by Eldan-Shamir shows exponential in 'n' gap between 1 and 2 hidden layers. Extend to k v/s k+1? k = O(1) v/s k = log(n)?
2. Restrict function to Boolean hypercube. Obtain gap results like in Boolean circuit complexity.

Theore ... very
natura ... ions
such th

1. f is
2. f is

Moreov ... with
the tor

Remark: More general versions, Approximation versions. n>=2 version using zonotopal norms.

Depth v/s size tradeoffs for ReLU DNNs

Restricting inputs to Boolean Hypercube (Mukherjee, Basu 2017):

1. 2 hidden layers always suffice: Any function on Boolean hypercube is a linear combination of the vertex-indicator functions. Each vertex indicator function can be implemented by a single ReLU gate.
2. Exponential lower bounds on ReLU DNN's of $O(n^c)$ depth to implement certain Boolean functions (for $c < 1/8$) under certain weight restrictions on first layer. Also implies some new Boolean circuit complexity results with LTF gates.

Discrete Geometry Techniques: Method of sign-rank and random restrictions.

Problems of interest for DNNs

- Expressiveness: What family of functions can one represent using DNNs?
- Efficiency: How many layers (depth) and vertices (size) needed to represent functions in the family?
- Training the network: Given architecture, data points (x,y), find weights for the "best fit" function.
- Generalization error: Rademacher complexity, VC dimension

Training Algorithm for ReLU-DNN(1,w)

Theorem (Arora, Basu, Mianjy, Mukherjee 2016): For. Let n, w be natural numbers, and $(x^1, y^1), \ldots, (x^D, y^D)$ a set of D data points in $R^n \times R$. There exists an algorithm that solves the following training problem to global optimality

$$\min\{\ |F(x^1) - y^1| + \ldots + |F(x^D) - y^D| : \ F \text{ in ReLU-DNN}(1,w)\ \}$$

The running time of the algorithm is $2^w D^{nw}$ poly(D,n,w).

Remark: More general convex loss functions can be handled

Training Algorithm for ReLU-DNN(1,w)

Characterization of ReLU(1,w) functions:

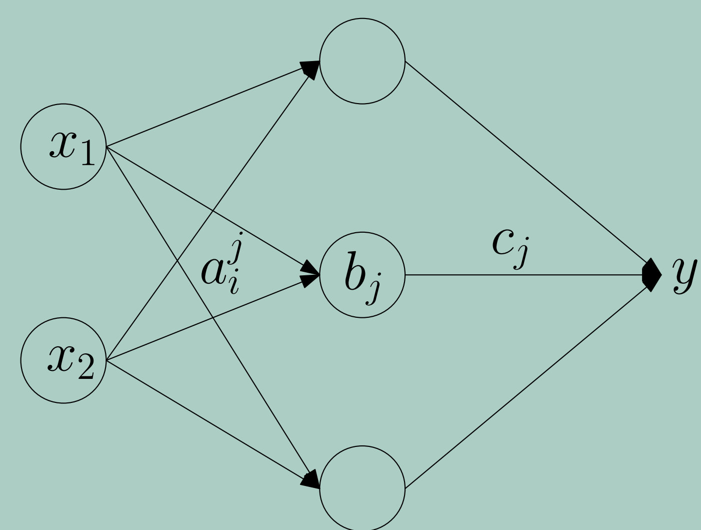
$$\max\{0, \langle p^1, x \rangle + q_1\} + \ldots + \max\{0, \langle p^k, x \rangle + q_k\}$$

$$- \max\{0, \langle n^1, x \rangle + h_1\} - \ldots - \max\{0, \langle n^s, x \rangle + h_s\}$$

Equivalently:

There is a hyperplane arrangement such that the function is affine linear in each cell of the hyperplane arrangement and whenever we "cross" a hyperplane in the arrangement, the value changes by the same linear function.

Training Algorithm for ReLU-DNN(1,w)



$$y = c_1 max\{0, \langle a^1, x\rangle + b_1\} + c_2 max\{0, \langle a^2, x\rangle + b_2\} + c_3 max\{0, \langle a^3, x\rangle + b_3\}$$

There … ion is affine linear in each cell of the hyperplane arrangement and whenever we "cross" a hyperplane in the arrangement, the value changes by the same linear function.

Training Algorithm for ReLU-DNN(1,w)

Characterization of ReLU(1,w) functions:

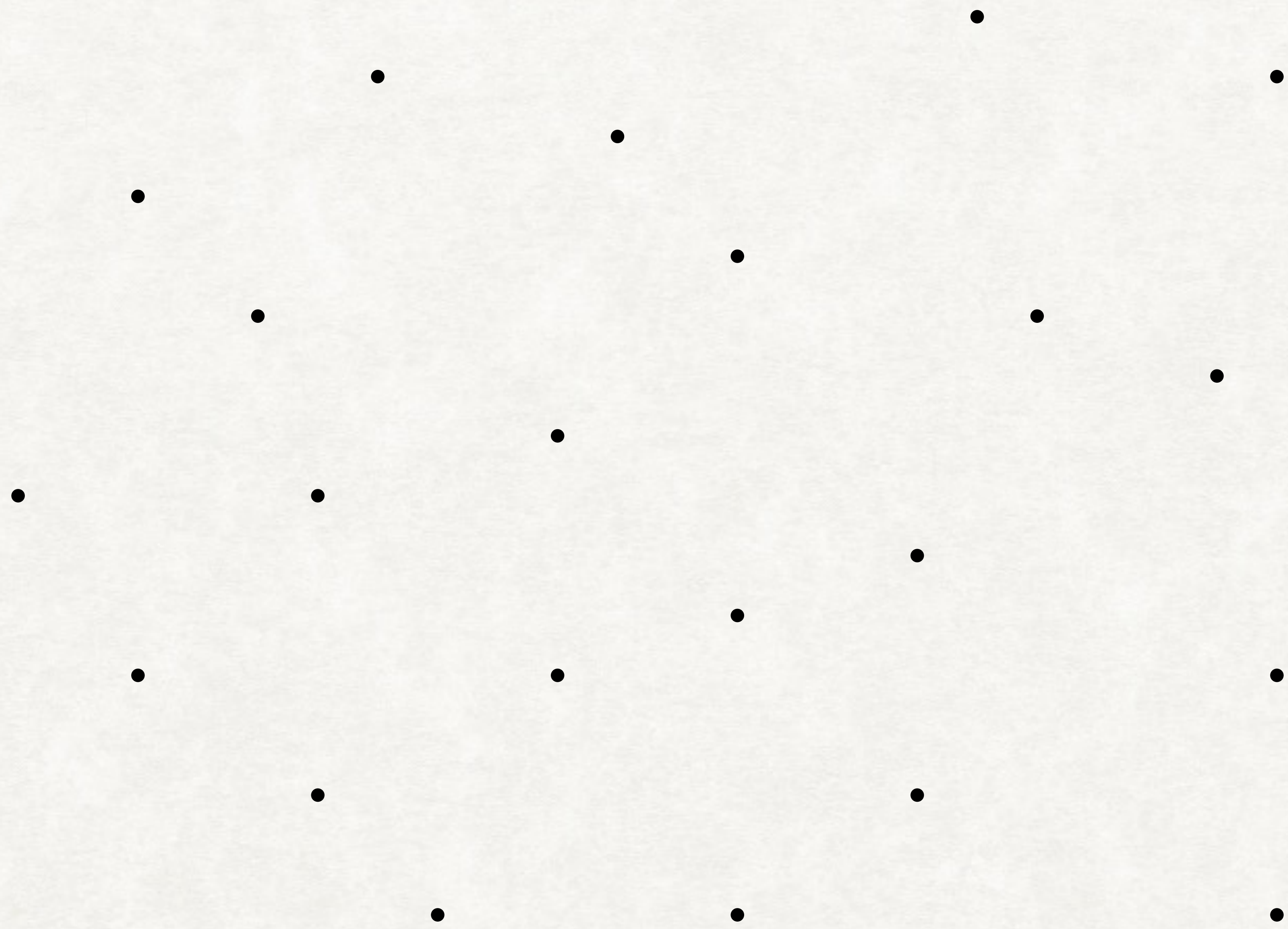
$$\max\{0, \langle p^1, x \rangle + q_1\} + \ldots + \max\{0, \langle p^k, x \rangle + q_k\}$$

$$- \max\{0, \langle n^1, x \rangle + h_1\} - \ldots - \max\{0, \langle n^s, x \rangle + h_s\}$$

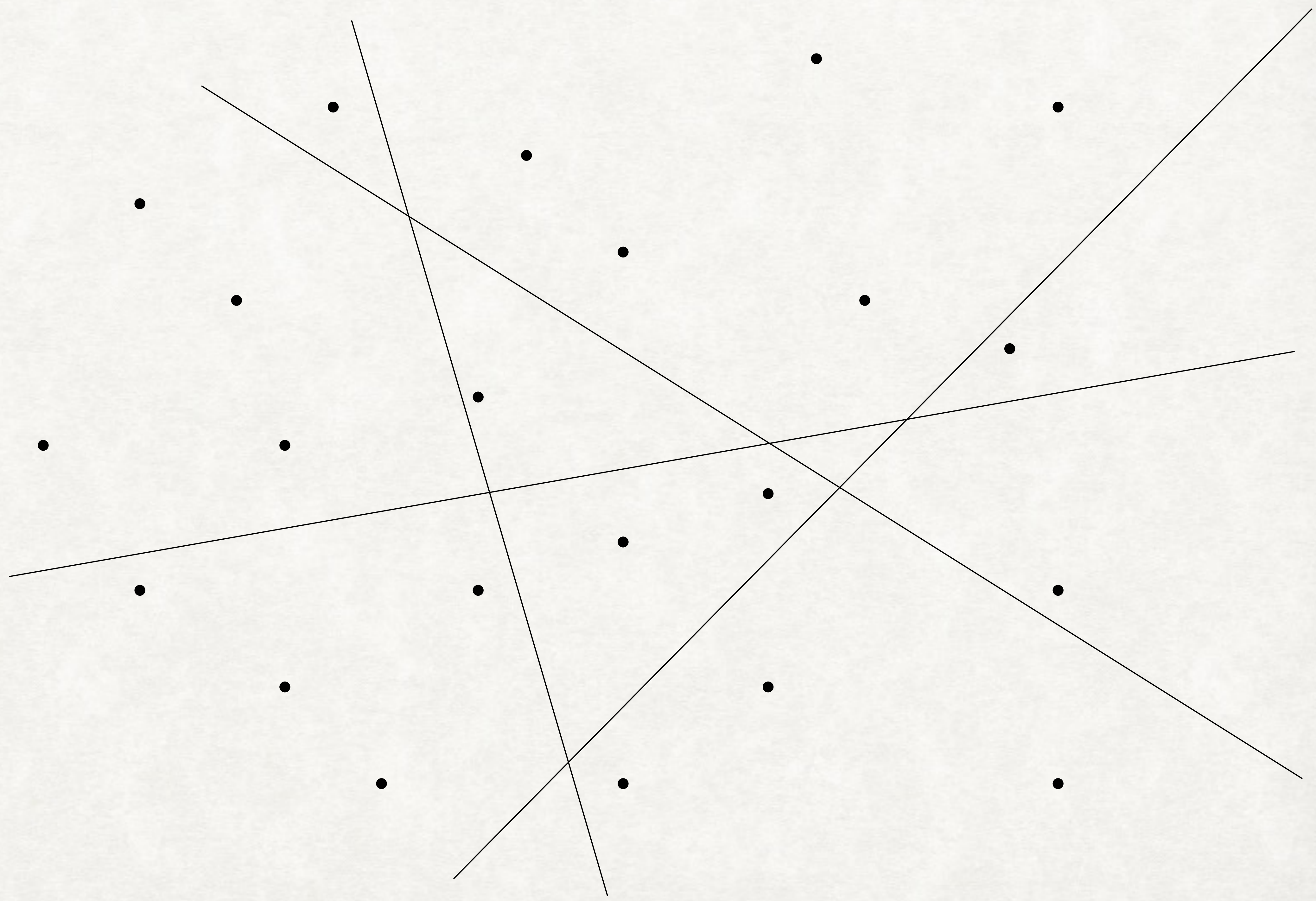
Equivalently:

There is a hyperplane arrangement such that the function is affine linear in each cell of the hyperplane arrangement and whenever we "cross" a hyperplane in the arrangement, the value changes by the same linear function.

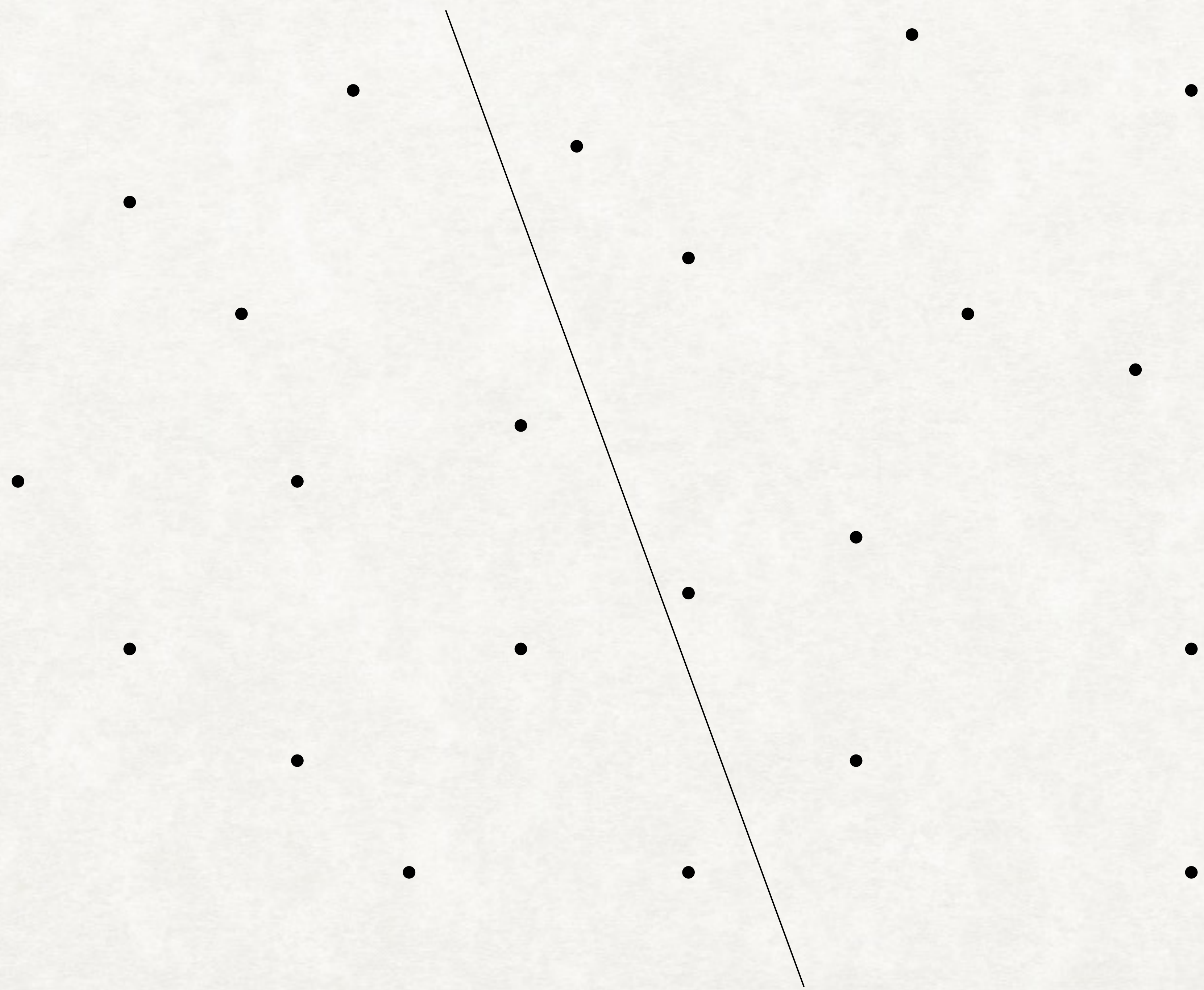
Training Algorithm for ReLU-DNN(1,w)



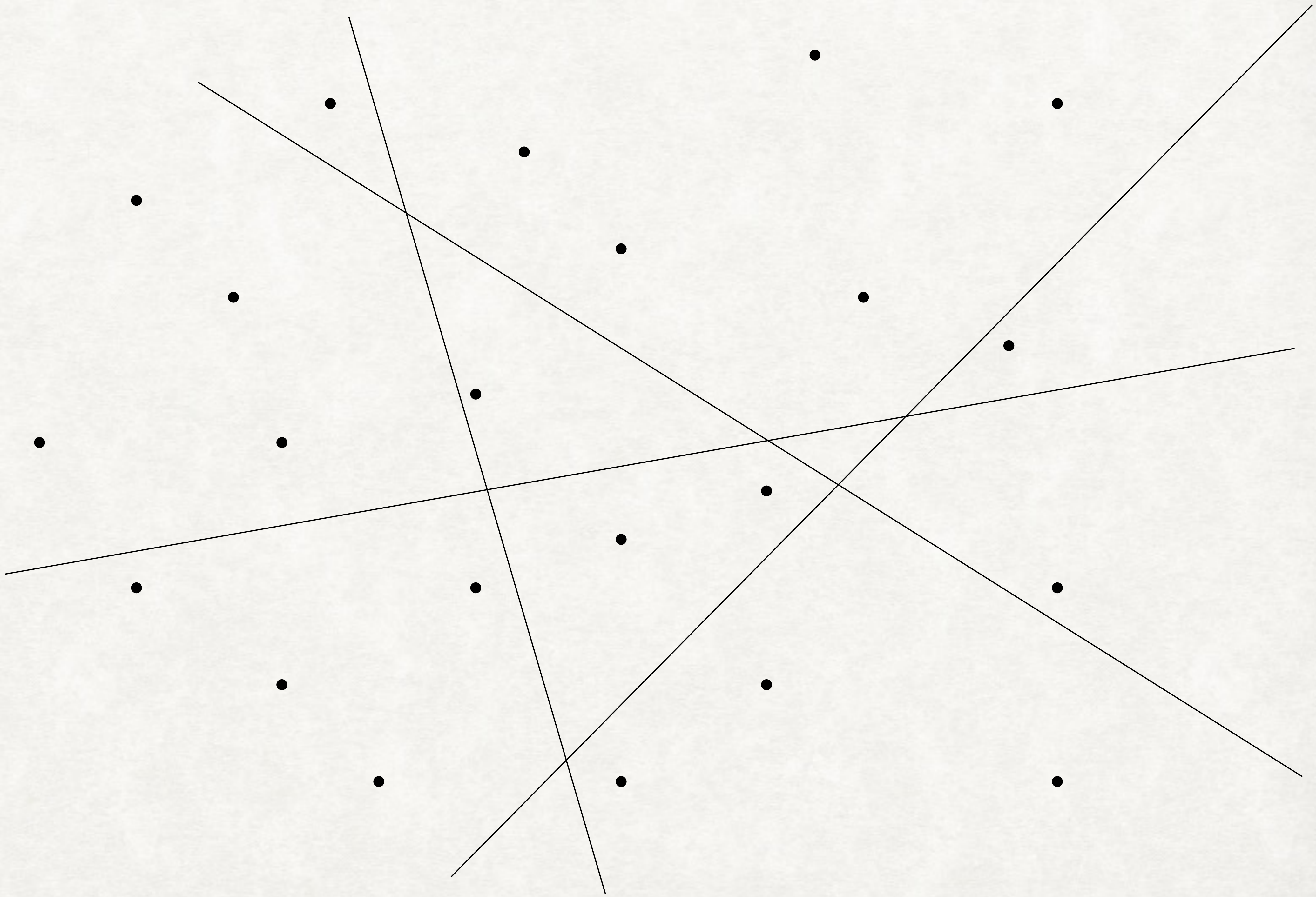
Training Algorithm for ReLU-DNN(1,w)



Training Algorithm for ReLU-DNN(1,w)



Training Algorithm for ReLU-DNN(1,w)



Training Algorithm for ReLU-DNN(1,w)

Theorem (Arora, Basu, Mianjy, Mukherjee 2016): For. Let n, w be natural numbers, and $(x^1, y^1), \ldots, (x^D, y^D)$ a set of D data points in $R^n \times R$. There exists an algorithm that solves the following training problem to global optimality

$$\min\{ |F(x^1) - y^1| + \ldots + |F(x^D) - y^D| : F \text{ in ReLU-DNN}(1,w) \}$$

The running time of the algorithm is $2^w D^{nw}$ poly(D,n,w).

Remark: More general convex loss functions can be handled

Training Algorithm for ReLU-DNN(1,w)

Theorem ... et n,
w be n ... of D
data po ... lves
the foll

min{ ... }

Open Questions

1. Exponential dependence on size 'w' necessary?
2. Training with 2 or more hidden layers.

The running time of the algorithm is $2^w D^{nw}$ poly(D,n,w).

Remark: More general convex loss functions can be handled

Thank you!

Questions/Comments/Answers?