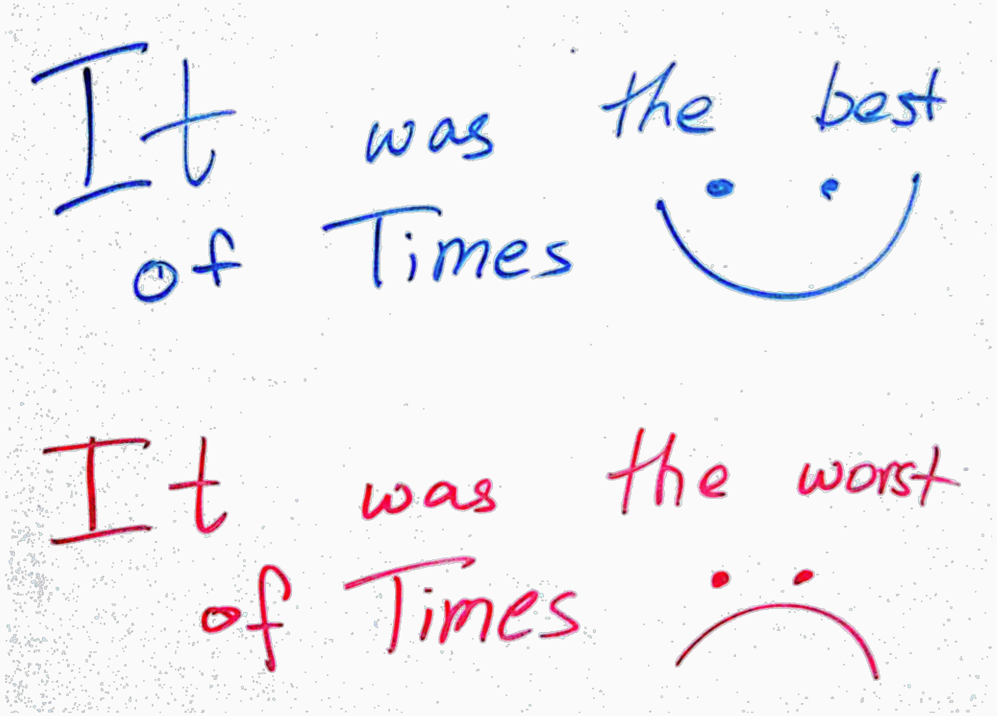
A Tale of Two Activations

Vivek Srikumar

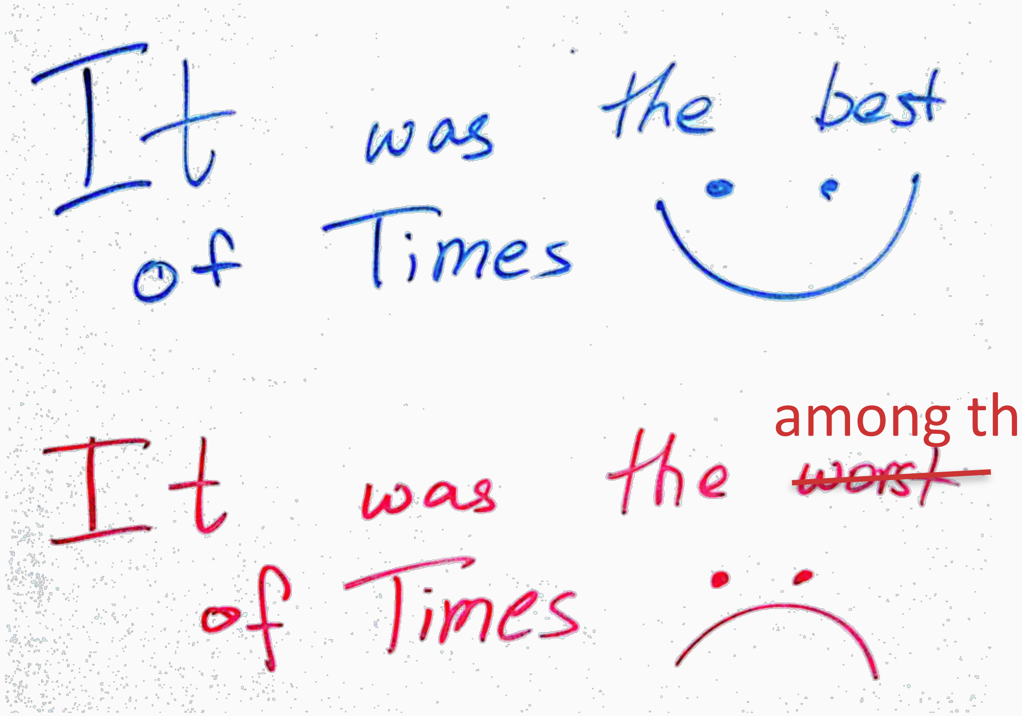
THE UNIVERSITY OF UTAH

Joint work with: *Xingyuan Pan, John Moeller, Sarathkrishna Swaminathan, Suresh Venkatasubramanian, Dustin Webb*

The neural network landscape



The neural network landscape



among the more confusing*

* caveats apply

Deep neural networks

Clear empirical successes

- Computer vision, speech processing, NLP

But why?

- Numerous design choices
- Many ideas in play: Nonlinear functions, more data, optimization ideas, regularization, representations, …

Deep neural networks

Clear empirical successes

- Computer vision, speech processing, NLP

But why?

- Numerous design choices
- Many ideas in play: Nonlinear functions, more data, optimization ideas, regularization, representations, …

What do each of these do?

Activation functions abound

- Identity
- Sign (or threshold)
- Sigmoids: Logistic, tanh, variants
- Polynomials
- Rectifiers and their variants
- Sinusoids, sinc, gaussian
- Pooling

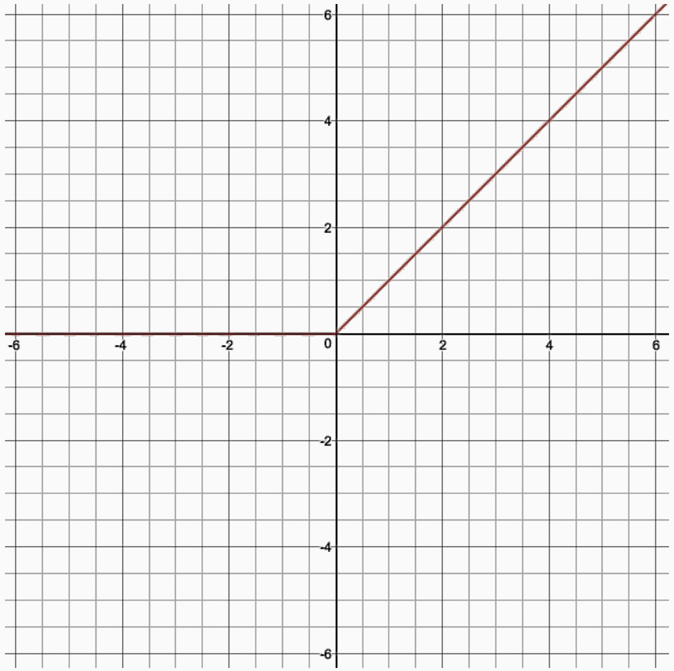
Motivating question: If I have to design a neural network for a new application, what should I use?

Effect on:

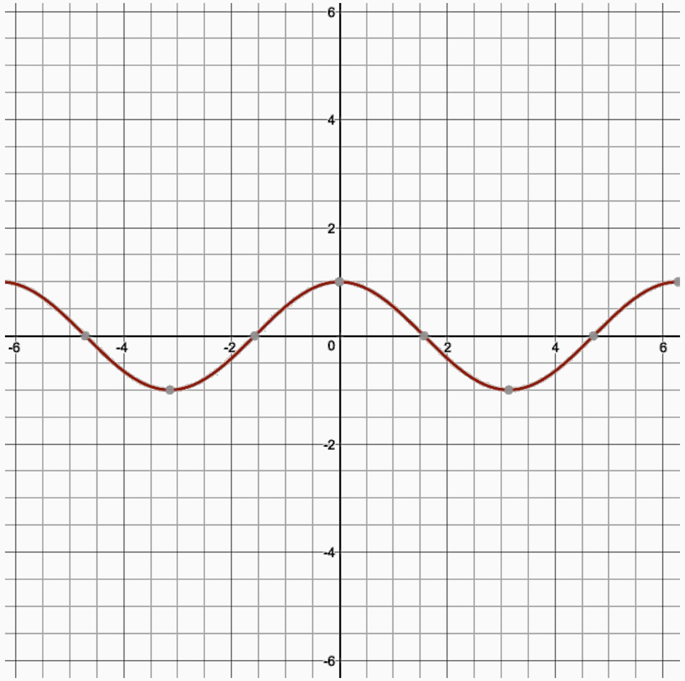
1. Expressiveness?
2. Sample complexity?

This talk: A tale of two activation functions

Rectified Linear Units

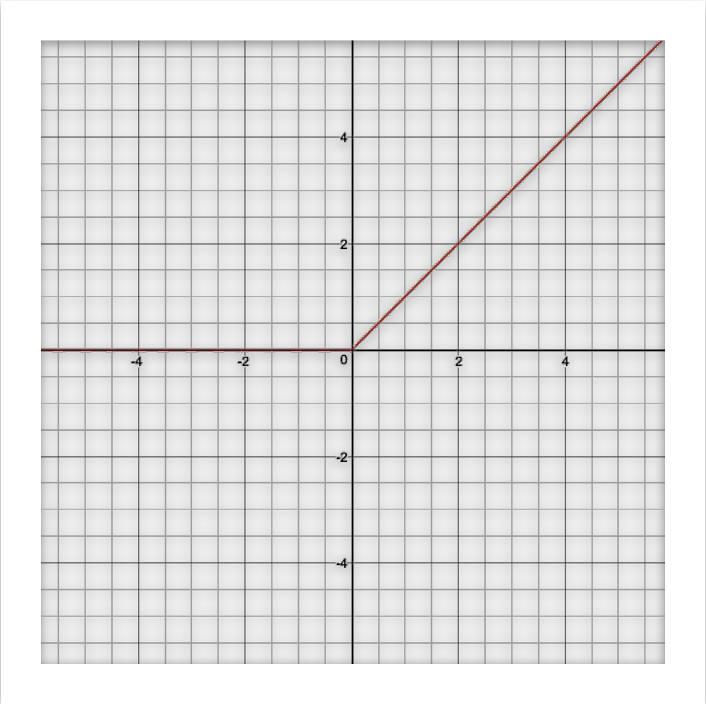


Cosine Neurons

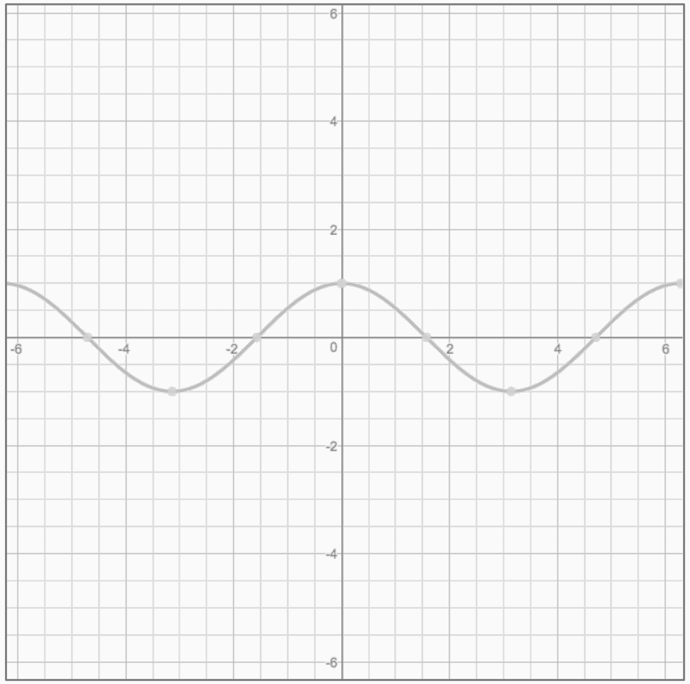


This talk: A tale of two activation functions

Rectified Linear Units



Cosine Neurons



Xingyuan Pan

Expressiveness of Rectifier Networks

ICML 2016

THE UNIVERSITY OF UTAH®

Expressiveness of Rectifier Networks: Outline

1. Rectifiers and Rectifier Networks

2. Decision boundaries of rectifier networks
   – Converting rectifier networks to threshold networks

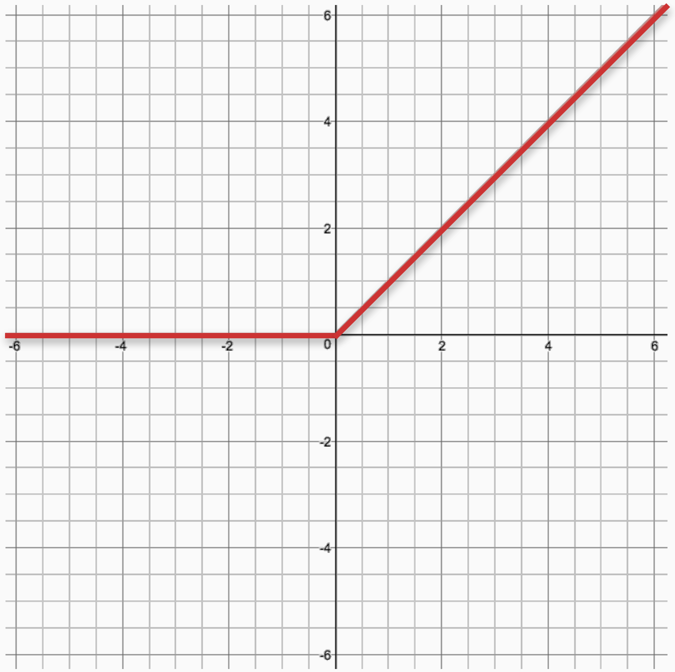
1. Thresholds to rectifiers and other open questions

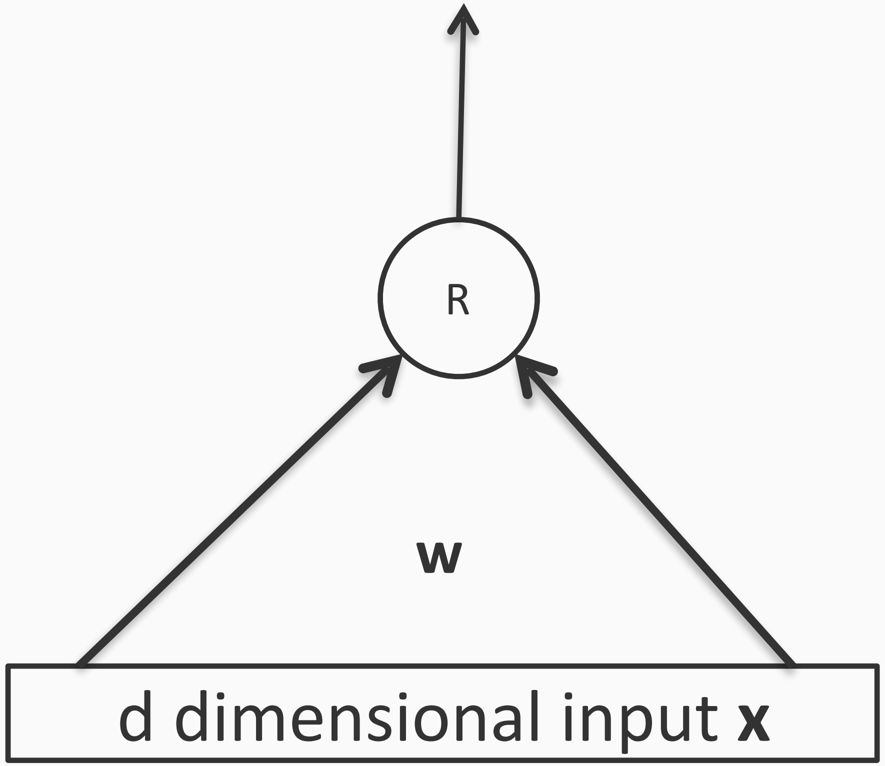
Rectifiers and Rectifier Networks

The rectified linear unit (ReLU)

Inputs: x, weights on incoming edges: w

Output: $\max\left(0, \mathbf{w}^T\mathbf{x}\right)$





ReLUs: A short (recent) history

- Used across speech, vision and NLP applications
  - [Nair & Hinton 2010, Glorot et al 2011, Krizhevsky et al 2012, Maas et al 2013…]

- A solution to an optimization concern [Glorot et al 2011]
  - ReLU activation is less affected by the vanishing gradient problem

- Encourages sparsity [Glorot et al 2011]
  - Because negative scores are clamped to zero

- Anectotally, faster to train: piecewise linear [Nair & Hinton 2010, Krizhevsky et al 2012]

Why do ReLUs work?

- State of the art in speech and vision applications
  - Often the default choice these days

- Computational advantages (i.e. optimization)
- Empirically better

But what functions do they express?

- Unexplored, unlike sigmoids and thresholds

What we know: Threshold and sigmoid functions

- Any continuous function can be approximated to arbitrary accuracy with one layer of sigmoid units [Cybenko 1989]

- Approximation error is insensitive to the choice of activation functions [DasGupta et al 1993]

- Two layer threshold networks can express any Boolean function

(All existential statements: In the worst case, these may need an exponential number of hidden units)

What we know: Threshold and sigmoid functions

- Any continuous function can be approximated to arbitrary accuracy with one layer of sigmoid units [Cybenko 1989]

- Appro ... s [DasGupta, 1993]

Can we make similar statements about rectifiers?

- Two layer threshold networks can express any Boolean function

(All existential statements: In the worst case, these may need an exponential number of hidden units)

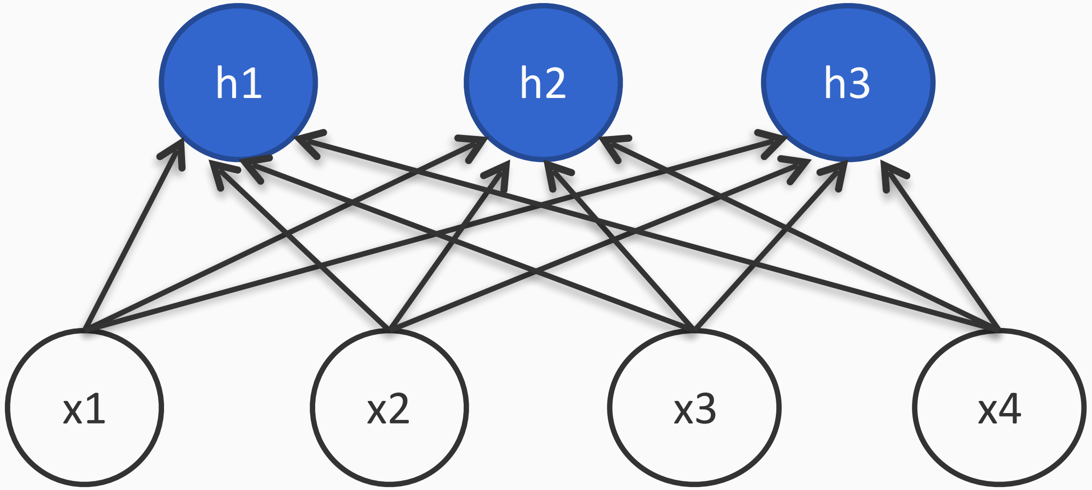
The setup: Two layer ReLU networks

The setup: Two layer ReLU networks

- Real valued inputs: $\mathbf{x}$

The setup: Two layer ReLU networks

- Hidden units (all rectifiers): $\mathrm{h}_1,\ \mathrm{h}_2\ldots$
- Real valued inputs: x

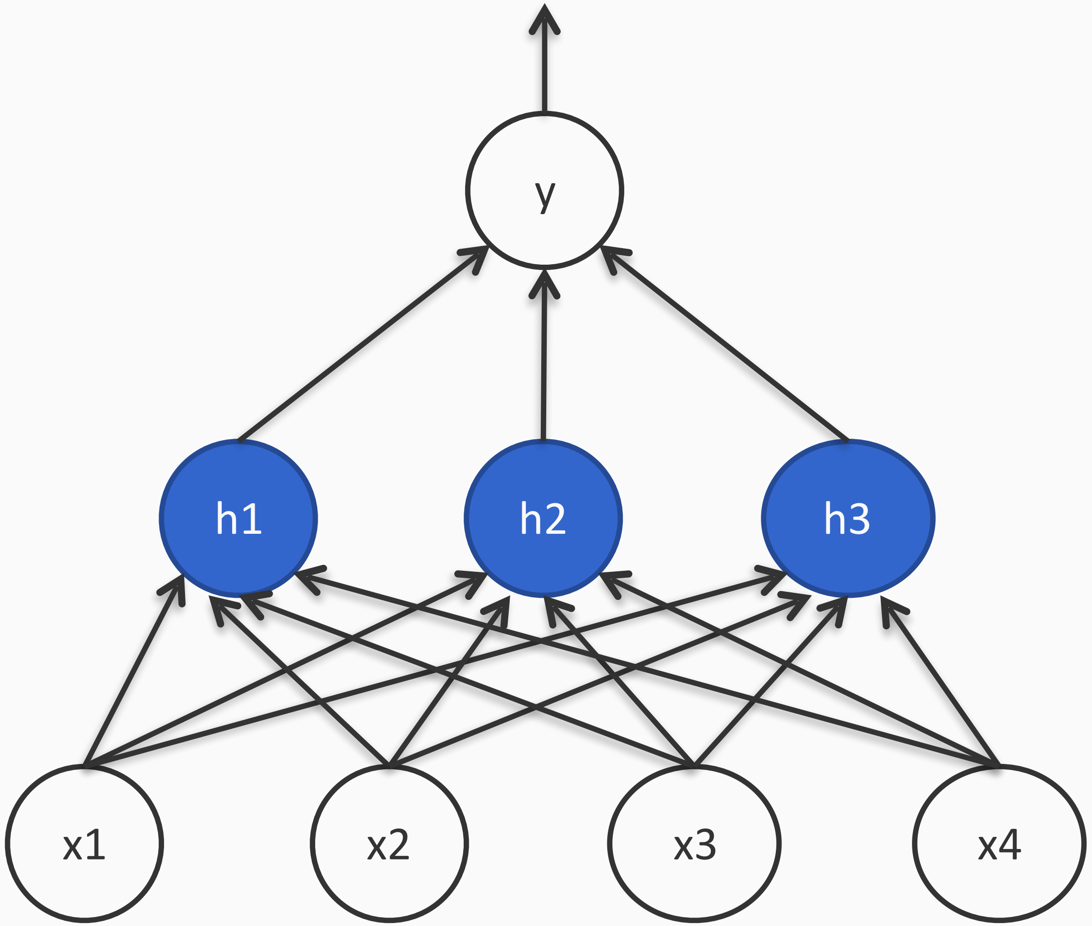


The setup: Two layer ReLU networks

- Output (threshold): y

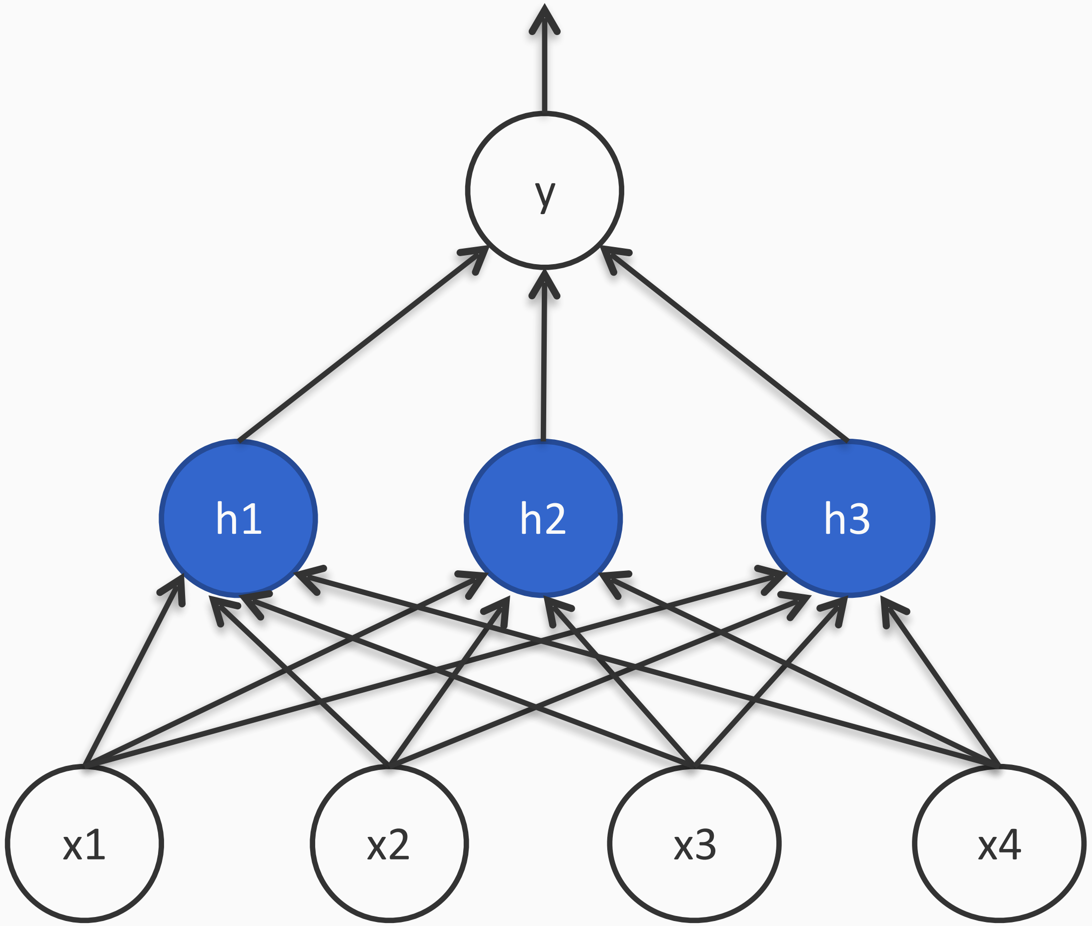
- Hidden units (all rectifiers): $\mathrm{h}_1,\ \mathrm{h}_2\ldots$

- Real valued inputs: x



The setup: Two layer ReLU networks

- Output (threshold): y
- Hidden units (all rectifiers): $\mathrm{h}_1,\ \mathrm{h}_2 \ldots$
- Real valued inputs: x



What decision functions can these kinds of networks represent?

The punchline

1. Two layer ReLU networks are equivalent to exponentially larger threshold networks

2. The converse is not always true Not every threshold network can be compressed into smaller rectifier networks

Decision boundaries of rectifier networks

Converting rectifier networks to threshold networks

Decision Boundaries: Threshold networks

How does a function slice up the input space into positive and negative regions?

A two layer threshold network with three hidden units

$$y = \operatorname{sgn}\left(w_0 + \sum_{k=1}^{3} w_k \operatorname{sgn}\left(\mathbf{v}_k^T \mathbf{x} + d_k\right)\right)$$

Decision Boundaries: Threshold networks

How does a function slice up the input space into positive and negative regions?

A two layer threshold network with three hidden units

$$y = \operatorname{sgn}\left(w_0 + \sum_{k=1}^{3} w_k \boxed{\operatorname{sgn}\left(\mathbf{v}_k^T \mathbf{x} + d_k\right)}\right)$$

Hidden units

Decision Boundaries: Threshold networks

How does a function slice up the input space into positive and negative regions?

A two layer threshold network with three hidden units

$$y = \boxed{\operatorname{sgn}\left(w_0 + \sum_{k=1}^{3} w_k \boxed{\operatorname{sgn}\left(\mathbf{v}_k^T \mathbf{x} + d_k\right)}\right)}$$

Hidden units

Output unit

Decision Boundaries: Threshold networks

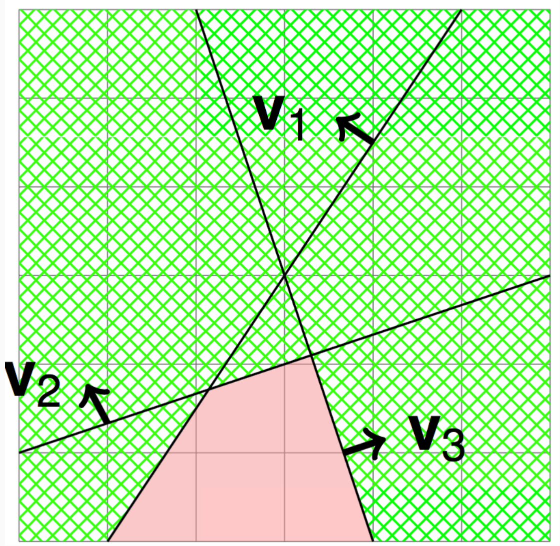
How does a function slice up the input space into positive and negative regions?

A two layer threshold network with three hidden units

$$y = \boxed{\mathrm{sgn}\left(w_0 + \sum_{k=1}^{3} w_k \boxed{\mathrm{sgn}\left(\mathbf{v}_k^T \mathbf{x} + d_k\right)}\right)}$$

Hidden units

Output unit



An example of a decision boundary

Decision Boundaries: Threshold networks

How does a function slice up the input space into positive and negative regions?

A two layer threshold network with n hidden units

$$y = \text{sgn}\left(w_0 + \sum_{k=1}^{n} w_k \text{sgn}\left(\mathbf{v}_k^T \mathbf{x} + d_k\right)\right)$$

Each hidden unit represents one hyperplane

- Parameterized by v_k and d_k
- Bisects the plane into two halfspaces

The output is an intersection (more generally, linear combination) of these half spaces

The general form of ReLU networks

$$y = \operatorname{sgn}\left(w_0 + \sum_{k=1}^{n} w_k \mathrm{R}\left(\mathbf{u}_k{}^T \mathbf{x} + d_k\right)\right)$$

The general form of ReLU networks

$$y = \operatorname{sgn}\left(w_0 + \sum_{k=1}^{n} w_k \mathrm{R}\left(\mathbf{u}_k{}^T \mathbf{x} + d_k\right)\right)$$

R(z) = max(0, z) $\Rightarrow$ c · R(z) = sgn(c) max(0, c· z)

The general form of ReLU networks

$$y = \operatorname{sgn}\left(w_0 + \sum_{k=1}^{n} w_k \mathrm{R}\left(\mathbf{u}_k{}^T \mathbf{x} + d_k\right)\right)$$

R(z) = max(0, z) $\Rightarrow$ c · R(z) = sgn(c) max(0, c· z)

That is, we can absorb the magnitude of w_k into the rectifier, leaving only the sign outside

$$y = \operatorname{sgn}\left(w_0 + \boxed{\text{All units where } w_k \text{ is positive}} - \boxed{\text{All units where } w_k \text{ is negative}}\right)$$

The general form of ReLU networks

$$y = \operatorname{sgn}\left(w_0 + \sum_{k=1}^{n} w_k \mathrm{R}\left({\mathbf{u}_k}^T \mathbf{x} + d_k\right)\right)$$

R(z) = max(0, z) $\Rightarrow$ c · R(z) = sgn(c) max(0, c· z)

That is, we can absorb the magnitude of w_k into the rectifier, leaving only the sign outside

$$y = \operatorname{sgn}\left(w_0 + \sum_{k \in P} \mathrm{R}\left({\mathbf{u}_k}^T \mathbf{x} + d_k\right) - \sum_{k \in N} \mathrm{R}\left({\mathbf{u}_k}^T \mathbf{x} + d_k\right)\right)$$

Coming up: Different configurations of ReLU networks

1. Two hidden units, all output weights one
2. n hidden units, all output weights one

3. Three hidden units, all output weights +1/-1
4. n hidden units, all output weights +1/-1

Decision Boundary: Rectifier Networks: Example 1

Two hidden rectifier units (denoted by R), only positive output weights

$$y = \operatorname{sgn}\left(-1 + \mathrm{R}\left(\mathbf{u_1}^T\mathbf{x} + b_1\right) + \mathrm{R}\left(\mathbf{u_2}^T\mathbf{x} + b_2\right)\right)$$

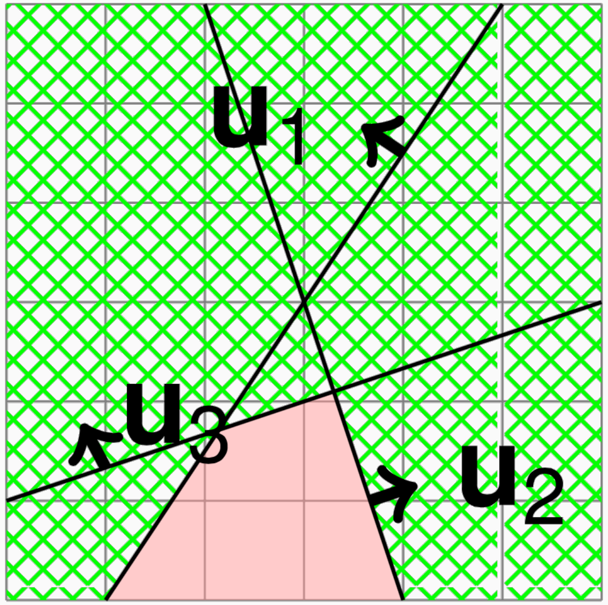
How does this function slice up the input space?

Decision Boundary: Rectifier Networks: Example 1

Two hidden rectifier units (denoted by R), only positive output weights

$$y = \operatorname{sgn}\left(-1 + \mathrm{R}\left(\mathbf{u}_1{}^T\mathbf{x} + b_1\right) + \mathrm{R}\left(\mathbf{u}_2{}^T\mathbf{x} + b_2\right)\right)$$

Suppose $\mathbf{u}_3 = \mathbf{u}_1 + \mathbf{u}_2$

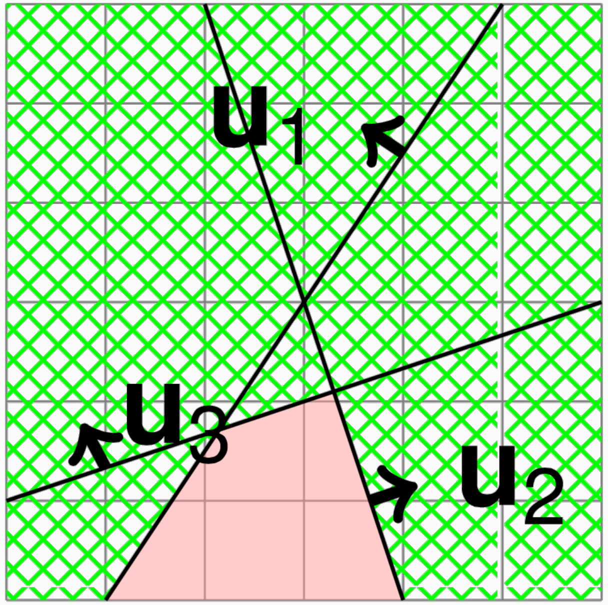


Decision Boundary: Rectifier Networks: Example 1

Two hidden rectifier units (denoted by R), only positive output weights

$$y = \text{sgn}\left(-1 + \text{R}\left(\mathbf{u_1}^T\mathbf{x} + b_1\right) + \text{R}\left(\mathbf{u_2}^T\mathbf{x} + b_2\right)\right)$$

Suppose $\mathbf{u}_3 = \mathbf{u}_1 + \mathbf{u}_2$



$$\begin{aligned}\phi_1(\mathbf{x}) &= -1 + \mathbf{u_1}^T x + b_1 \\ \phi_2(\mathbf{x}) &= -1 + \mathbf{u_2}^T x + b_2 \\ \phi_3(\mathbf{x}) &= -1 + \mathbf{u_3}^T x + b_1 + b_2\end{aligned}$$

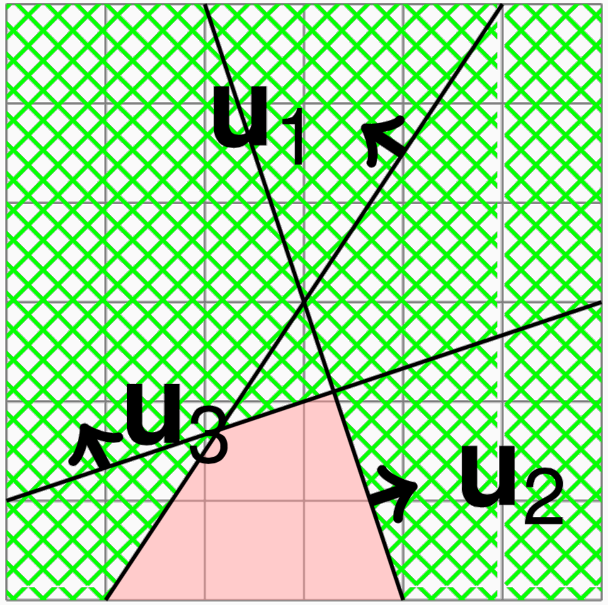
The output is +1 if, and only if, at least one of these three ϕ's are positive

Decision Boundary: Rectifier Networks: Example 1

Two hidden rectifier units (denoted by R), only positive output weights

$$y = \operatorname{sgn}\left(-1 + \mathrm{R}\left({\mathbf{u}_1}^T\mathbf{x} + b_1\right) + \mathrm{R}\left({\mathbf{u}_2}^T\mathbf{x} + b_2\right)\right)$$

Suppose $\mathbf{u}_3 = \mathbf{u}_1 + \mathbf{u}_2$



Take away message

Two hidden ReLUs with positive output weights express the same function as *a specific* threshold network with three hidden units

Decision Boundary: Rectifier Network: Example 2

n hidden rectifier units (denoted by R), only positive output weights

$$y = \operatorname{sgn}\left(w_0 + \sum_{k=1}^{n} \mathrm{R}\left(\mathbf{u}_k{}^T \mathbf{x} + b_k\right)\right)$$

Decision Boundary: Rectifier Network: Example 2

n hidden rectifier units (denoted by R), only positive output weights

$$y = \mathrm{sgn}\left(w_0 + \sum_{k=1}^{n} \mathrm{R}\left(\mathbf{u}_k{}^T\mathbf{x} + b_k\right)\right)$$

The output is positive if, and only if, there ***exists*** a subset S_1 of {1, 2, ⋯, n}, such that

$$w_0 + \sum_{k \in S_1} \mathbf{u}_k{}^T\mathbf{x} + b_k \geq 0$$

Decision Boundary: Rectifier Network: Example 2

n hidden rectifier units (denoted by R), only positive output weights

$$y = \operatorname{sgn}\left(w_0 + \sum_{k=1}^{n} \mathrm{R}\left({\color{red}\mathbf{u}_k}^T \mathbf{x} + b_k\right)\right)$$

The output is positive if, and only if, there ***<u>exists</u>*** a subset S_1 of {1, 2, $\cdots$, n}, such that

$$w_0 + \sum_{k \in S_1} {\color{red}\mathbf{u}_k}^T \mathbf{x} + b_k \geq 0$$

Each choice of S_1 gives a different hyperplane. There are 2^n such choices

Which of them is the right one? The output layer is a disjunction layer

Decision Boundary: Rectifier Network: Example 2

n hidden rectifier units (denoted by R), only positive output weights

$$y = \mathrm{sgn}\left(w_0 + \sum_{k=1}^{n} \mathrm{R}\left(\mathbf{u}_k{}^T \mathbf{x} + b_k\right)\right)$$

Take away message

A two layer ReLU network with n hidden positive units expresses the same function as a fixed threshold function with 2^n hidden units

Decision Boundary: Rectifier Network: Example 3

Three hidden rectifier units (denoted by R), positive and negative output weights

$$y = \operatorname{sgn}\left(-1 + \mathrm{R}\left({u_1}^T\mathbf{x} + b_1\right) - \mathrm{R}\left({u_2}^T\mathbf{x} + b_2\right) - \mathrm{R}\left({u_2}^T\mathbf{x} + b_3\right)\right)$$

Decision Boundary: Rectifier Network: Example 3

Three hidden rectifier units (denoted by R), positive and negative output weights

$$y = \mathrm{sgn}\left(-1 + \mathrm{R}\left(\underbrace{\boxed{{u_1}^T\mathbf{x} + b_1}}_{a_1(\mathrm{x})}\right) - \mathrm{R}\left(\underbrace{\boxed{{u_2}^T\mathbf{x} + b_2}}_{a_2(\mathrm{x})}\right) - \mathrm{R}\left(\underbrace{\boxed{{u_2}^T\mathbf{x} + b_3}}_{a_3(\mathrm{x})}\right)\right)$$

Decision Boundary: Rectifier Network: Example 3

Three hidden rectifier units (denoted by R), positive and negative output weights

$$y = \operatorname{sgn}\left(-1 + \mathrm{R}\left(a_1(\mathbf{x})\right) - \mathrm{R}\left(a_2(\mathbf{x})\right) - \mathrm{R}\left(a_3(\mathbf{x})\right)\right)$$

Decision Boundary: Rectifier Network: Example 3

Three hidden rectifier units (denoted by R), positive and negative output weights

$$y = \mathrm{sgn}\left(-1 + \mathrm{R}\left(a_1(\mathbf{x})\right) - \mathrm{R}\left(a_2(\mathbf{x})\right) - \mathrm{R}\left(a_3(\mathbf{x})\right)\right)$$

The output is positive if, and only if, one of these sets of inequalities are all true

$$\begin{aligned} w_0 &\geq 0 \\ w_0 - a_2(\mathbf{x}) &\geq 0 \\ w_0 - a_3(\mathbf{x}) &\geq 0 \\ w_0 - a_2(\mathbf{x}) - a_3(\mathbf{x}) &\geq 0 \end{aligned}$$

$$\begin{aligned} w_0 + a_1(\mathbf{x}) &\geq 0 \\ w_0 + a_1(\mathbf{x}) - a_2(\mathbf{x}) &\geq 0 \\ w_0 + a_1(\mathbf{x}) - a_3(\mathbf{x}) &\geq 0 \\ w_0 + a_1(\mathbf{x}) - a_2(\mathbf{x}) - a_3(\mathbf{x}) &\geq 0 \end{aligned}$$

Decision Boundary: Rectifier Network: Example 3

Three hidden rectifier units (denoted by R), positive and negative output weights

$$y = \operatorname{sgn}\left(-1 + \mathrm{R}\left(a_1(\mathbf{x})\right) - \mathrm{R}\left(a_2(\mathbf{x})\right) - \mathrm{R}\left(a_3(\mathbf{x})\right)\right)$$

The output is positive if, and only if, one of these sets of inequalities are all true

$$\begin{aligned} w_0 &\geq 0 \\ w_0 - a_2(\mathbf{x}) &\geq 0 \\ w_0 - a_3(\mathbf{x}) &\geq 0 \\ w_0 - a_2(\mathbf{x}) - a_3(\mathbf{x}) &\geq 0 \end{aligned}$$

$$\begin{aligned} w_0 + a_1(\mathbf{x}) &\geq 0 \\ w_0 + a_1(\mathbf{x}) - a_2(\mathbf{x}) &\geq 0 \\ w_0 + a_1(\mathbf{x}) - a_3(\mathbf{x}) &\geq 0 \\ w_0 + a_1(\mathbf{x}) - a_2(\mathbf{x}) - a_3(\mathbf{x}) &\geq 0 \end{aligned}$$

One block for each subset of the positive units (here, a_1)

Decision Boundary: Rectifier Network: Example 3

Three hidden rectifier units (denoted by R), positive and negative output weights

$$y = \operatorname{sgn}\left(-1 + \mathrm{R}\left(a_1(\mathbf{x})\right) - \mathrm{R}\left(a_2(\mathbf{x})\right) - \mathrm{R}\left(a_3(\mathbf{x})\right)\right)$$

The output is positive if, and only if, one of these sets of inequalities are all true

$$\begin{aligned} w_0 &\geq 0 \\ w_0 - a_2(\mathbf{x}) &\geq 0 \\ w_0 - a_3(\mathbf{x}) &\geq 0 \\ w_0 - a_2(\mathbf{x}) - a_3(\mathbf{x}) &\geq 0 \end{aligned}$$

$$\begin{aligned} w_0 + a_1(\mathbf{x}) &\geq 0 \\ w_0 + a_1(\mathbf{x}) - a_2(\mathbf{x}) &\geq 0 \\ w_0 + a_1(\mathbf{x}) - a_3(\mathbf{x}) &\geq 0 \\ w_0 + a_1(\mathbf{x}) - a_2(\mathbf{x}) - a_3(\mathbf{x}) &\geq 0 \end{aligned}$$

One block for each subset of the positive units (here, a_1)

Decision Boundary of a general Rectifier Network

$$y = \operatorname{sgn}\left(w_0 + \sum_{k \in P} \mathrm{R}\left(\mathbf{u}_k{}^T \mathbf{x} + d_k\right) - \sum_{k \in N} \mathrm{R}\left(\mathbf{u}_k{}^T \mathbf{x} + d_k\right)\right)$$

Decision Boundary of a general Rectifier Network

$$y = \operatorname{sgn}\left(w_0 + \sum_{k \in P} \mathrm{R}\left(\mathbf{u}_k{}^T \mathbf{x} + d_k\right) - \sum_{k \in N} \mathrm{R}\left(\mathbf{u}_k{}^T \mathbf{x} + d_k\right) \right)$$

Theorem: The following statements are equivalent:

Decision Boundary of a general Rectifier Network

$$y = \operatorname{sgn}\left(w_0 + \sum_{k \in P} \mathrm{R}\left(\mathbf{u}_k{}^T \mathbf{x} + d_k\right) - \sum_{k \in N} \mathrm{R}\left(\mathbf{u}_k{}^T \mathbf{x} + d_k\right) \right)$$

Theorem: The following statements are equivalent:

1. The value of y is positive

Decision Boundary of a general Rectifier Network

$$y = \operatorname{sgn}\left(w_0 + \sum_{k \in P} \mathrm{R}\left(\mathbf{u}_k{}^T \mathbf{x} + d_k\right) - \sum_{k \in N} \mathrm{R}\left(\mathbf{u}_k{}^T \mathbf{x} + d_k\right)\right)$$

Theorem: The following statements are equivalent:

1. The value of y is positive

2. There ***exists*** a subset S_1 of P such that, for ***every*** subset S_2 of N, we have

$$w_0 + \left(\sum_{k \in S_1} \mathbf{u}_k{}^T \mathbf{x} + d_k\right) - \left(\sum_{k \in S_2} \mathbf{u}_k{}^T \mathbf{x} + d_k\right) \geq 0$$

Decision Boundary of a general Rectifier Network

$$y = \operatorname{sgn}\left(w_0 + \sum_{k \in P} \mathrm{R}\left(\mathbf{u}_k{}^T \mathbf{x} + d_k\right) - \sum_{k \in N} \mathrm{R}\left(\mathbf{u}_k{}^T \mathbf{x} + d_k\right)\right)$$

Theorem: The following statements are equivalent:

1. The value of y is positive

2. There ***exists*** a subset S_1 of P such that, for ***every*** subset S_2 of N, we have

$$w_0 + \left(\sum_{k \in S_1} \mathbf{u}_k{}^T \mathbf{x} + d_k\right) - \left(\sum_{k \in S_2} \mathbf{u}_k{}^T \mathbf{x} + d_k\right) \geq 0$$

3. For every subset S_2 of N, we can find a subset S_1 of P such that the above condition holds.

Let’s look at the previous example again

Three hidden rectifier units (denoted by R), positive and negative output weights

$$y = \operatorname{sgn}\left(-1 + \mathrm{R}\left(a_1(\mathbf{x})\right) - \mathrm{R}\left(a_2(\mathbf{x})\right) - \mathrm{R}\left(a_3(\mathbf{x})\right)\right)$$

“There ***exists*** *a subset S_1 of P such that, for* ***every*** *subset S_2 of N....”*

Let’s look at the previous example again

Three hidden rectifier units (denoted by R), positive and negative output weights

$$y = \operatorname{sgn}\left(-1 + \mathrm{R}\left(a_1(\mathbf{x})\right) - \mathrm{R}\left(a_2(\mathbf{x})\right) - \mathrm{R}\left(a_3(\mathbf{x})\right)\right)$$

$$\boxed{\begin{aligned} w_0 &\geq 0 \\ w_0 - a_2(\mathbf{x}) &\geq 0 \\ w_0 - a_3(\mathbf{x}) &\geq 0 \\ w_0 - a_2(\mathbf{x}) - a_3(\mathbf{x}) &\geq 0 \end{aligned}}$$

The chosen subset of positive units is empty

*"There **exists** a subset S_1 of P such that, for **every** subset S_2 of N…."*

Let's look at the previous example again

Three hidden rectifier units (denoted by R), positive and negative output weights

$$y = \operatorname{sgn}\left(-1 + \mathrm{R}\left(a_1(\mathbf{x})\right) - \mathrm{R}\left(a_2(\mathbf{x})\right) - \mathrm{R}\left(a_3(\mathbf{x})\right)\right)$$

$$\begin{aligned} w_0 &\geq 0 \\ w_0 - a_2(\mathbf{x}) &\geq 0 \\ w_0 - a_3(\mathbf{x}) &\geq 0 \\ w_0 - a_2(\mathbf{x}) - a_3(\mathbf{x}) &\geq 0 \end{aligned}$$

The chosen subset of positive units is empty

$$\begin{aligned} w_0 + a_1(\mathbf{x}) &\geq 0 \\ w_0 + a_1(\mathbf{x}) - a_2(\mathbf{x}) &\geq 0 \\ w_0 + a_1(\mathbf{x}) - a_3(\mathbf{x}) &\geq 0 \\ w_0 + a_1(\mathbf{x}) - a_2(\mathbf{x}) - a_3(\mathbf{x}) &\geq 0 \end{aligned}$$

The chosen subset of positive units contains a_1

*"There **exists** a subset S_1 of P such that, for **every** subset S_2 of N…."*

Let's look at the previous example again

Three hidden rectifier units (denoted by R), positive and negative output weights

$$y = \operatorname{sgn}\left(-1 + \mathrm{R}\left(a_1(\mathbf{x})\right) - \mathrm{R}\left(a_2(\mathbf{x})\right) - \mathrm{R}\left(a_3(\mathbf{x})\right)\right)$$

The output is positive if, and only if, one of these sets of inequalities are all true

$$\begin{aligned} w_0 &\geq 0 \\ w_0 - a_2(\mathbf{x}) &\geq 0 \\ w_0 - a_3(\mathbf{x}) &\geq 0 \\ w_0 - a_2(\mathbf{x}) - a_3(\mathbf{x}) &\geq 0 \end{aligned}$$

The chosen subset of positive units is empty

$$\begin{aligned} w_0 + a_1(\mathbf{x}) &\geq 0 \\ w_0 + a_1(\mathbf{x}) - a_2(\mathbf{x}) &\geq 0 \\ w_0 + a_1(\mathbf{x}) - a_3(\mathbf{x}) &\geq 0 \\ w_0 + a_1(\mathbf{x}) - a_2(\mathbf{x}) - a_3(\mathbf{x}) &\geq 0 \end{aligned}$$

The chosen subset of positive units contains a_1

How many hidden threshold units?

$$y = \mathrm{sgn}\left(w_0 + \sum_{k \in P} \mathrm{R}\left(\mathbf{u}_k{}^T \mathbf{x} + d_k\right) - \sum_{k \in N} \mathrm{R}\left(\mathbf{u}_k{}^T \mathbf{x} + d_k\right)\right)$$

- The output is positive if there ***exists*** a subset S_1 of P such that, for ***every*** subset S_2 of N, we have

$$w_0 + \left(\sum_{k \in S_1} \mathbf{u}_k{}^T \mathbf{x} + d_k\right) - \left(\sum_{k \in S_2} \mathbf{u}_k{}^T \mathbf{x} + d_k\right) \geq 0$$

How many hidden threshold units?

$$y = \operatorname{sgn}\left(w_0 + \sum_{k \in P} \mathrm{R}\left(\mathbf{u}_k{}^T \mathbf{x} + d_k\right) - \sum_{k \in N} \mathrm{R}\left(\mathbf{u}_k{}^T \mathbf{x} + d_k\right)\right)$$

- The output is positive if there ***exists*** a subset S_1 of P such that, for ***every*** subset S_2 of N, we have

$$w_0 + \left(\sum_{k \in S_1} \mathbf{u}_k{}^T \mathbf{x} + d_k\right) - \left(\sum_{k \in S_2} \mathbf{u}_k{}^T \mathbf{x} + d_k\right) \geq 0$$

- A different hyperplane for each choice of S_1 and S_2

- There are 2^n such choices if there are n hidden ReLU units in all

Expressiveness of Rectifier Networks

A two layer network with n ReLU hidden units expresses the same Boolean function as a threshold network with 2^n hidden units (no longer two layered)

Expressiveness of Rectifier Networks

A two layer network with n ReLU hidden units expresses the same Boolean function as a threshold network with 2^n hidden units (no longer two layered)

Proof by construction

A tight bound. There are rectifier networks that cannot be represented by any fewer than exponential threshold units

Thresholds to rectifiers and other open questions

From thresholds to rectifiers

Can all threshold networks be represented reduced to rectifier networks with a logarithmically fewer units?

From thresholds to rectifiers

Can all threshold networks be represented reduced to rectifier networks with a logarithmically fewer units?

Generally: No.

Proof by counter example that needs n ReLU units

$$y = \operatorname{sgn}\left(n - 1 + \sum_{i=1}^{n} \operatorname{sgn}(x_i)\right)$$

From thresholds to rectifiers

Can all threshold networks be represented reduced to rectifier networks with a logarithmically fewer units?

Generally: No.

Proof by counter example that needs n ReLU units

$$y = \operatorname{sgn}\left(n - 1 + \sum_{i=1}^{n} \operatorname{sgn}(x_i)\right)$$

But the paper characterizes sufficient conditions for where this can be done

Expressiveness of ReLUs: Summary

- Rectifier networks are widely used, but not well studied

- We showed:
  - **Good news**: The decision boundary of a two layer rectifier network is exactly the same as if we were using an exponential number of threshold units
    - A tight bound

  - **Bad news**: In general, not all large threshold networks can be represented using fewer ReLU units

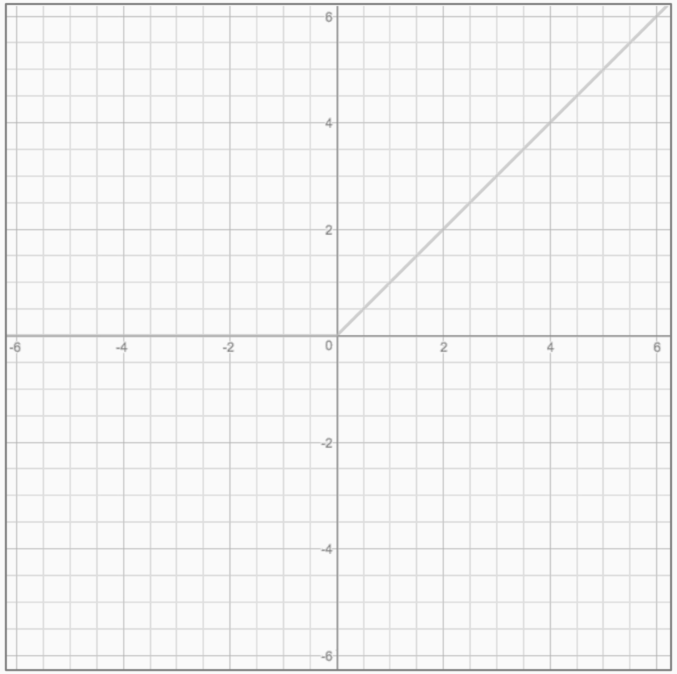
Questions looking ahead

Using rectifier hidden units effectively explores the space of larger threshold networks

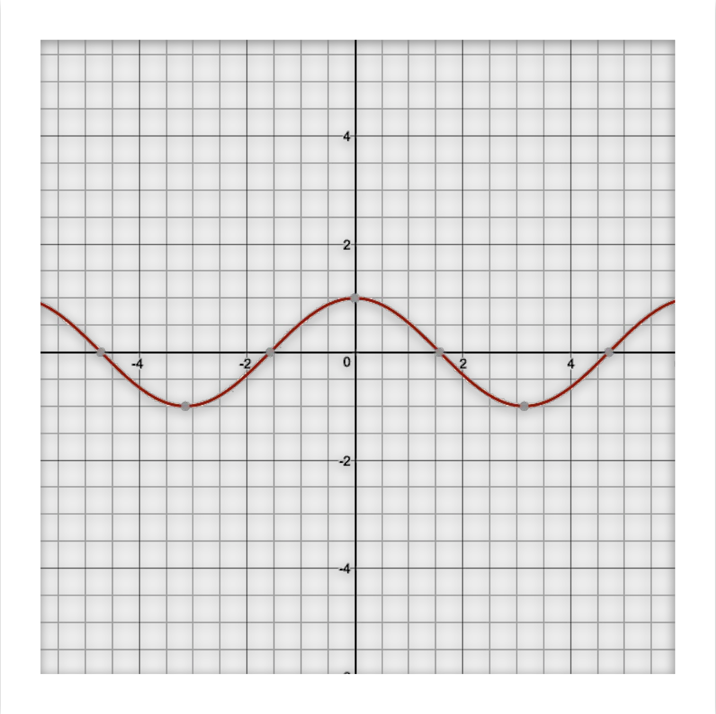
- How does this affect learnability?
- More hidden layers complicate the analysis. Needs more work

This talk: A tale of two activation functions

Rectified Linear Units

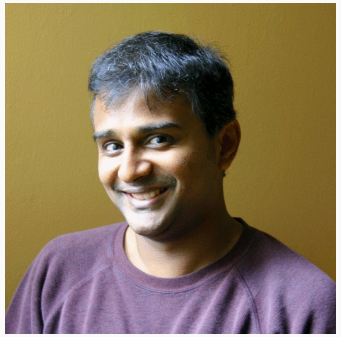


Cosine Neurons




John Moeller


Sarath
Swaminathan
(not his picture)


Suresh
Venkatasubramanian


Dustin Webb

Continuous Kernel Learning

ECML 2016

The setup: Two layer cosine networks

The setup: Two layer cosine networks

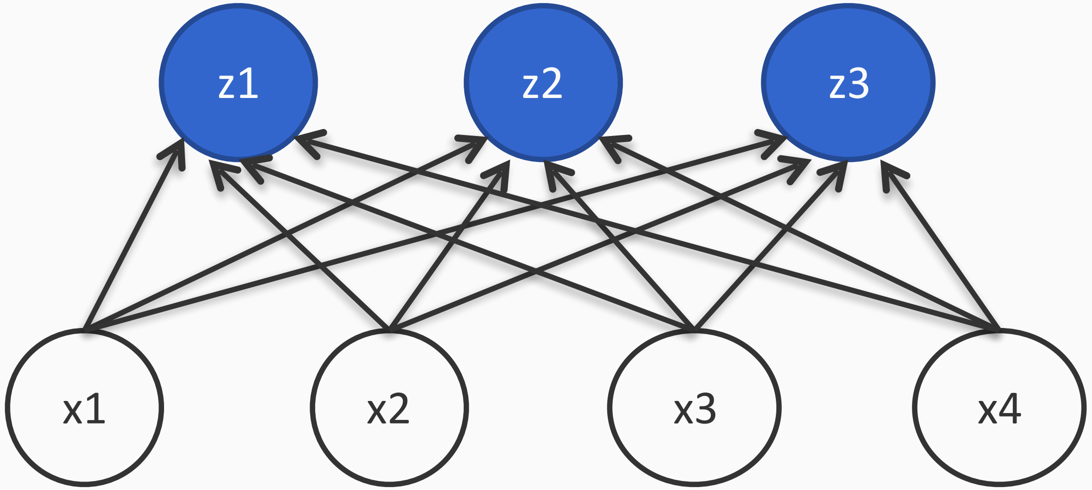
- Real valued inputs: x

The setup: Two layer cosine networks

- Hidden units: $z_1,\ z_2 \ldots$

$$\boxed{\text{all}\cos(\cdot)}$$

- Real valued inputs: x



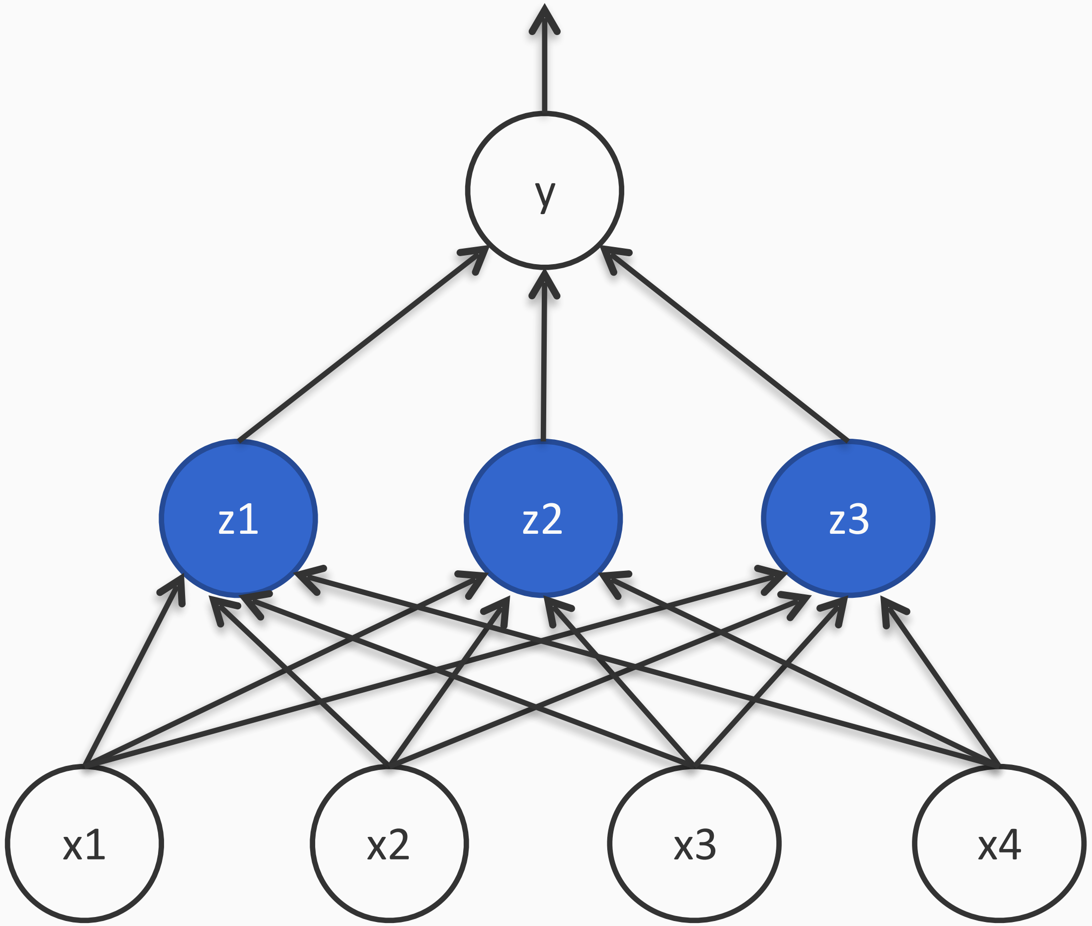
The setup: Two layer cosine networks

- Output (threshold): y

- Hidden units: $\mathrm{z}_1,\ \mathrm{z}_2\ldots$

$\boxed{\text{all}\cos(\cdot)}$

- Real valued inputs: x



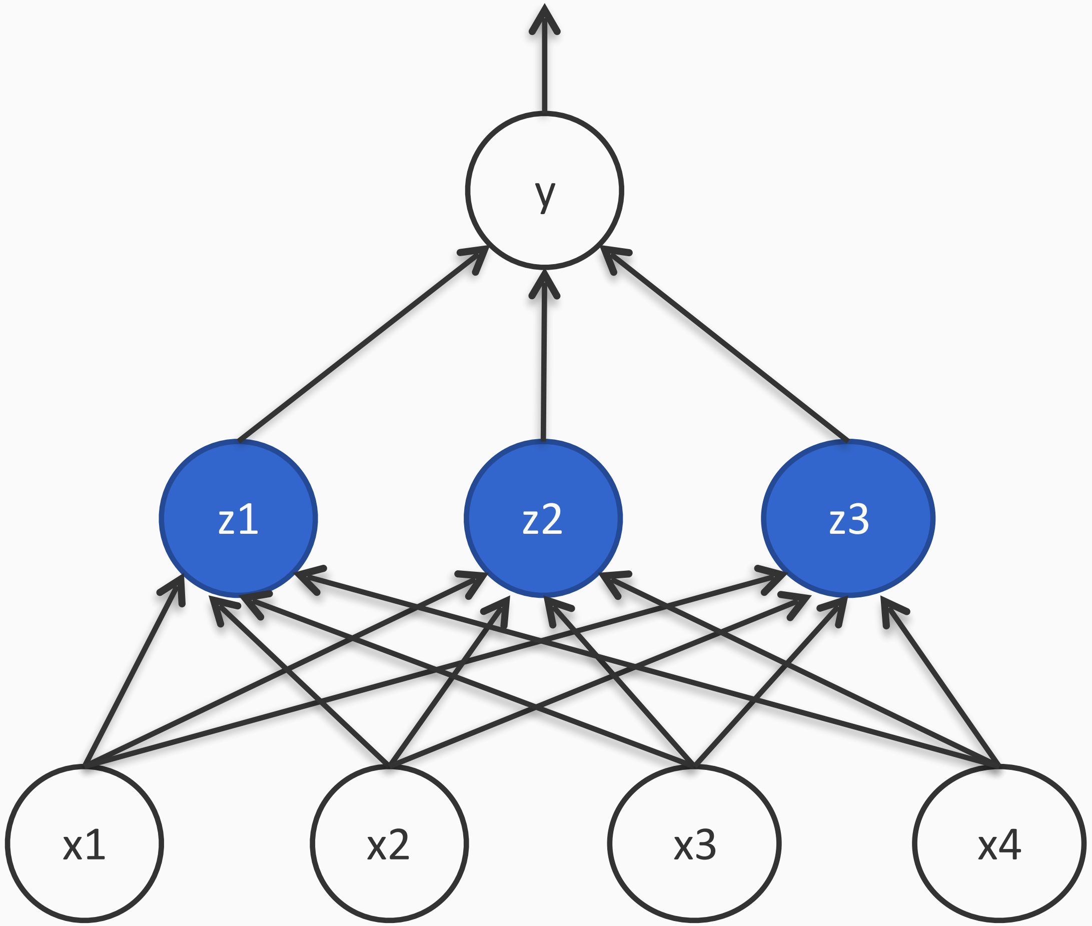
The setup: Two layer cosine networks

- Output (threshold): y

- Hidden units: $\mathrm{z}_1,\ \mathrm{z}_2 \ldots$

$$\boxed{\text{all}\cos(\cdot)}$$

- Real valued inputs: x



What can we say about these networks?

Continuous Kernel Learning: Outline

- How did the cosine come about?
  - Connections to kernels
- Capacity of the cosine
- Experiments

How did the cosine come about?

Bochner's theorem [Bochner 1959]

Every shift-invariant continuous positive definite kernel is the Fourier transform of a unique probability distribution

Bochner's theorem [Bochner 1959]

Every shift-invariant continuous positive definite kernel is the Fourier transform of a unique probability distribution

$$\kappa(\mathbf{x}, \mathbf{x}') \underset{\text{Fourier}}{\Leftrightarrow} p(\omega)$$

Bochner's theorem [Bochner 1959]

Every shift-invariant continuous positive definite kernel is the Fourier transform of a unique probability distribution

$$\boxed{\kappa(\mathbf{x}, \mathbf{x}')} \underset{\text{Fourier}}{\Leftrightarrow} p(\omega)$$

Bochner's theorem [Bochner 1959]

Every shift-invariant continuous positive definite kernel is the Fourier transform of a unique probability distribution

$$\boxed{\kappa(\mathbf{x}, \mathbf{x}')} \underset{\text{Fourier}}{\Leftrightarrow} \boxed{p(\omega)}$$

Bochner's theorem [Bochner 1959]

Every shift-invariant continuous positive definite kernel is the Fourier transform of a unique probability distribution

$$\boxed{\kappa(\mathbf{x}, \mathbf{x}')} \underset{\text{Fourier}}{\Leftrightarrow} \boxed{p(\omega)}$$

Kernels are equivalent to distributions

Kernels = Distributions via Fourier transforms

Bochner

$$\kappa(\mathbf{x}, \mathbf{x}') = \int e^{i\mathbf{w}^T(\mathbf{x}-\mathbf{x}')} p(\mathbf{w}) d\mathbf{w}$$

Kernels = Distributions via Fourier transforms

Bochner

$$\kappa(\mathbf{x}, \mathbf{x}') = \int e^{i\mathbf{w}^T(\mathbf{x}-\mathbf{x}')} p(\mathbf{w}) d\mathbf{w}$$

But this is the definition of a expectation!

$$\kappa(\mathbf{x}, \mathbf{x}') = E_{\mathbf{w}}\left[e^{i\mathbf{w}^T(\mathbf{x}-\mathbf{x}')}\right]$$

Random Fourier Features [Rahimi and Recht 2007]

$$\kappa(\mathbf{x}, \mathbf{x}') = E_{\mathbf{w}}\left[e^{i\mathbf{w}^T(\mathbf{x}-\mathbf{x}')}\right]$$

Replace expectation with samples

$$\kappa(\mathbf{x}, \mathbf{x}') \approx \sum_k e^{i\mathbf{w}_k^T(\mathbf{x}-\mathbf{x}')}$$

Random Fourier Features [Rahimi and Recht 2007]

$$\kappa(\mathbf{x}, \mathbf{x}') = E_{\mathbf{w}} \left[e^{i\mathbf{w}^T(\mathbf{x}-\mathbf{x}')} \right]$$

Replace expectation with samples

$$\kappa(\mathbf{x}, \mathbf{x}') \approx \sum_k e^{i\mathbf{w}_k^T(\mathbf{x}-\mathbf{x}')}$$

We can drop the imaginary part of RHS because LHS is real

Random Fourier Features

$$\kappa(\mathbf{x}, \mathbf{x}') = E_{\mathbf{w}}\left[e^{i\mathbf{w}^T(\mathbf{x}-\mathbf{x}')}\right]$$

Replace expectation with samples

$$\kappa(\mathbf{x}, \mathbf{x}') \approx \sum_k \cos(\mathbf{w}_k^T\mathbf{x} + b_k)\cos(\mathbf{w}_k^T\mathbf{x}' + b_k)$$

Random Fourier Features: The recipe

Suppose we know the kernel we want to compute.

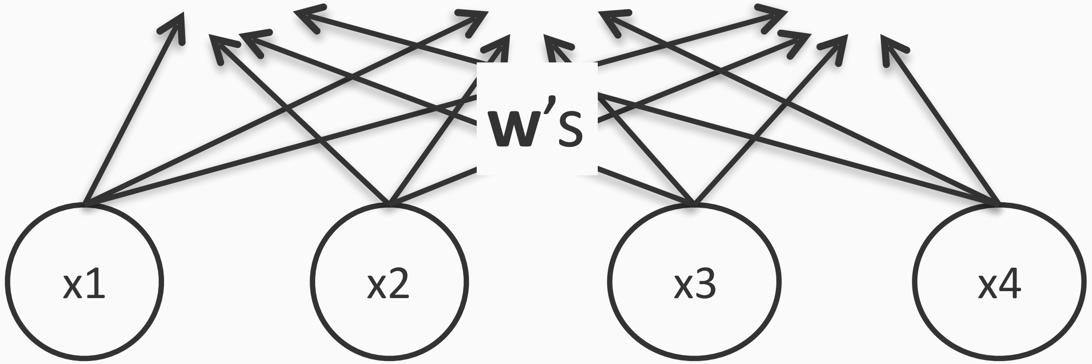
Calculate the probability distribution corresponding to it

Random Fourier Features: The recipe

Suppose we know the kernel we want to compute.

Calculate the probability distribution corresponding to it

Draw many samples from this distribution (**w**)



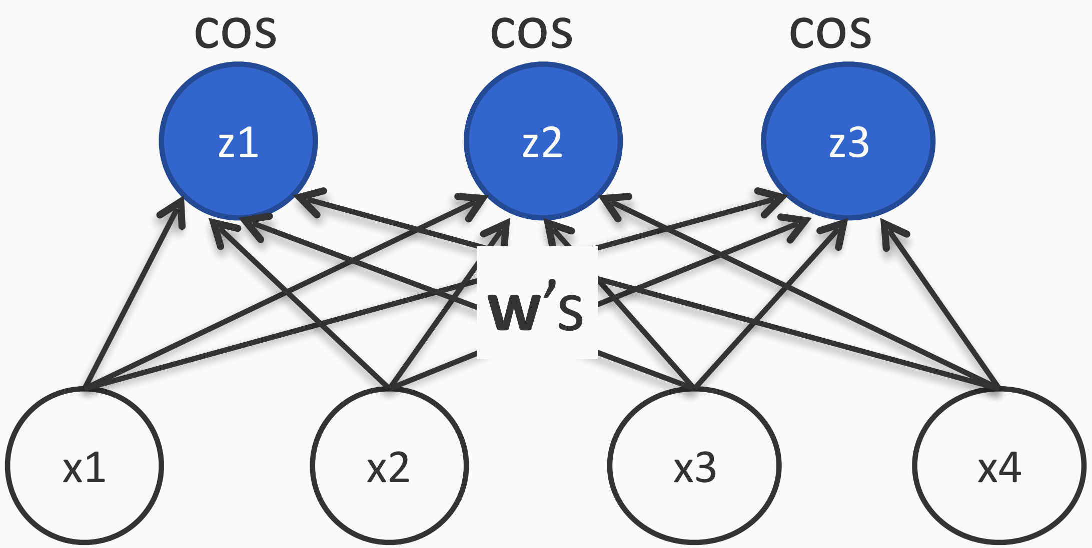
Random Fourier Features: The recipe

Suppose we know the kernel we want to compute.

Calculate the probability distribution corresponding to it

Draw many samples from this distribution (**w**)

Construct the representation **z**



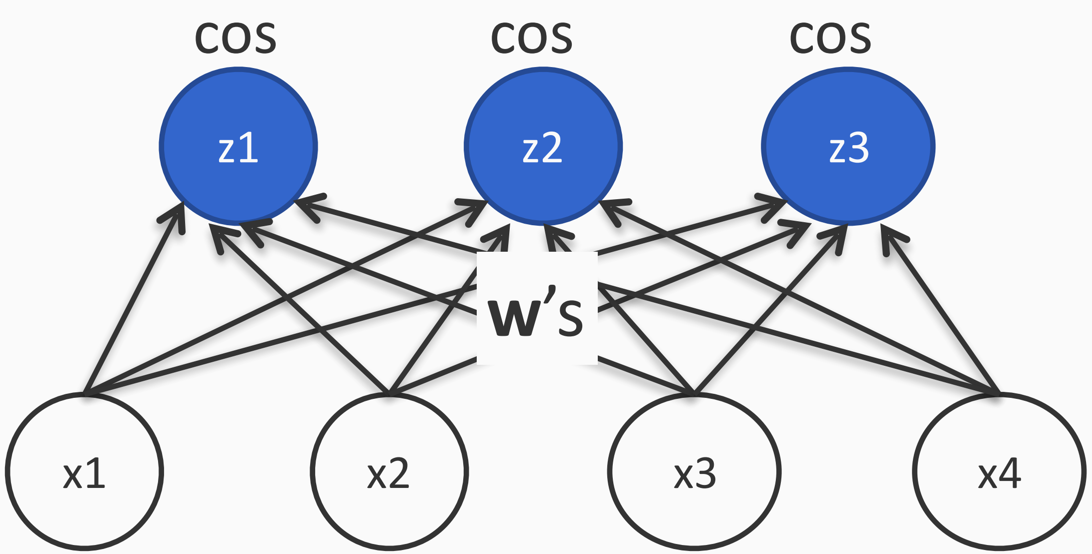
Random Fourier Features: The recipe

Suppose we know the kernel we want to compute.

Calculate the probability distribution corresponding to it

Draw many samples from this distribution (**w**)

Construct the representation **z**



Dot product of the **z**'s approximates the kernel

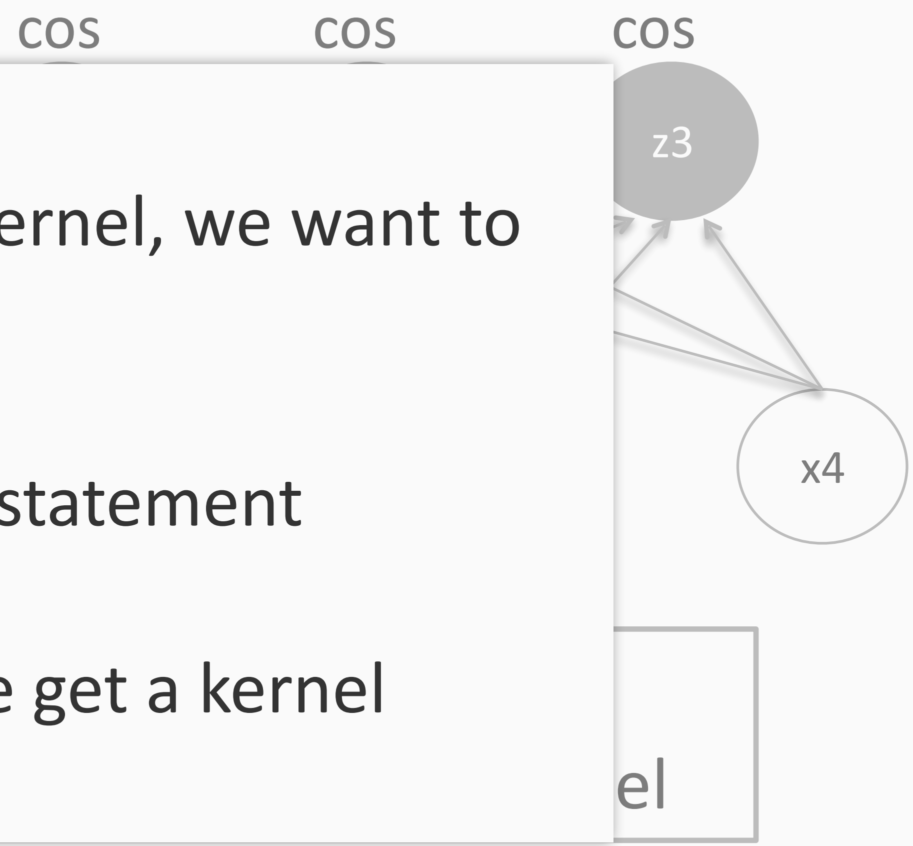
Random Fourier Features: The recipe

Suppose we know the kernel we want
to com

Calcula
corresp

Draw n
distribu

Constr



Random Fourier Features: We have a kernel, we want to approximate it

Bochner’s theorem is an if, and only, if statement

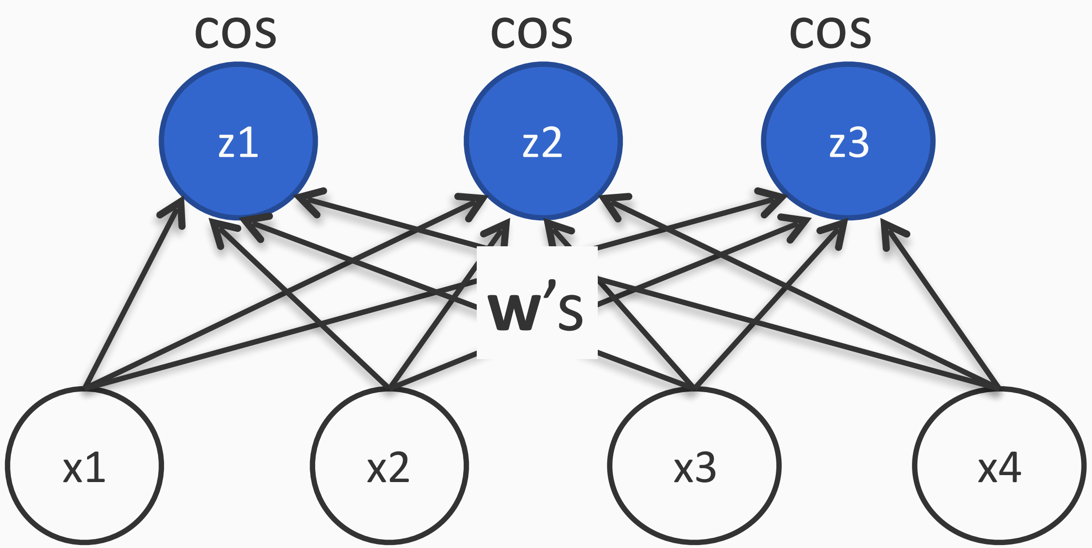
If we have **w**’s and construct the **z**’s, we get a kernel

Changing the weights gives us a new kernel

Every **w** corresponds to a probability distribution

Every probability distribution corresponds to a shift-invariant kernel

Change the w = change the kernel



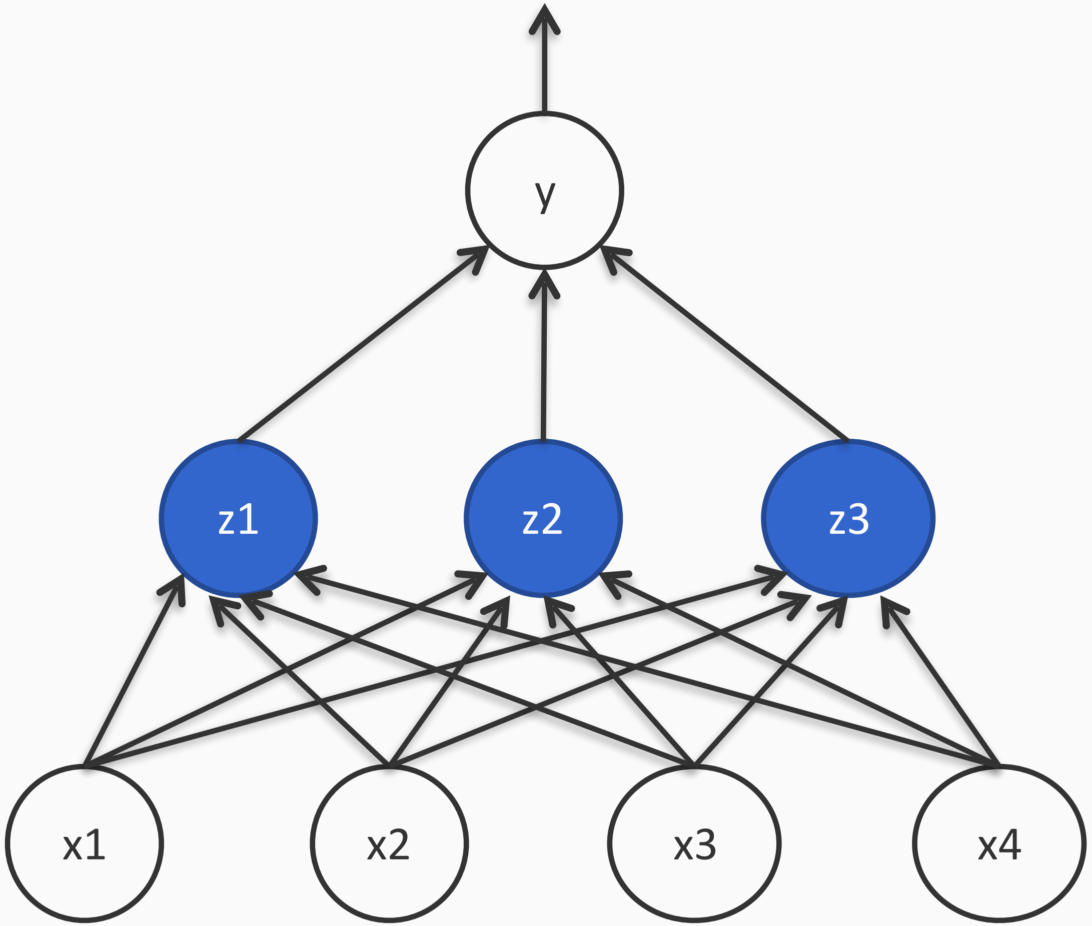
Searching over the space of kernels

We can

1. Learn the **w**'s to get a new representation of the inputs

2. Learn a linear classifier on top of this new representation

A two layer neural network whose hidden layer consists of cosine activation functions



Related: arc-cosine kernels [Yang et al 2013], Gaussian processes [Wilson et al 2015] as neural networks

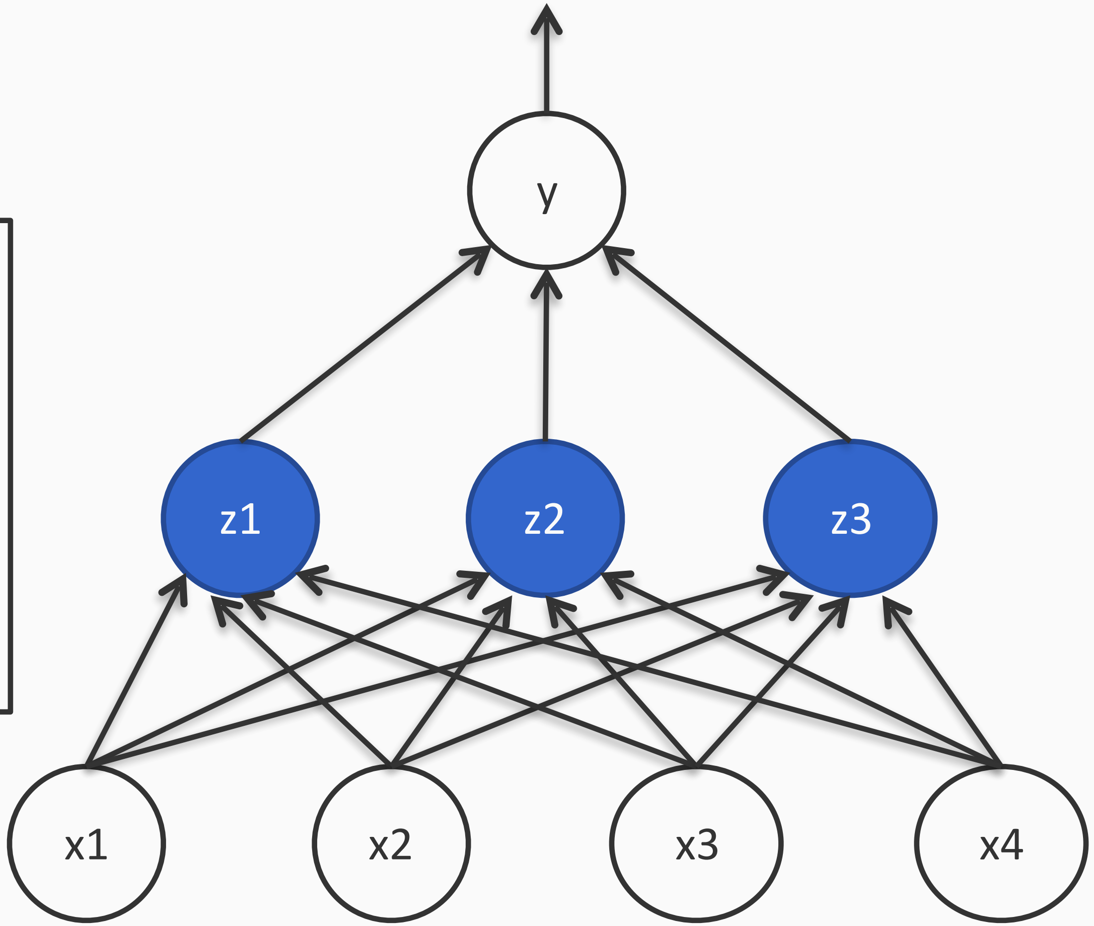
Searching over the space of kernels

We can

Use backpropagation to search over the entire space of shift invariant kernels!

Continuous kernel learning

A two layer neural network whose hidden layer consists of cosine activation functions



Related: arc-cosine kernels [Yang et al 2013], Gaussian processes [Wilson et al 2015] as neural networks

THE UNIVERSITY OF UTAH

Searching over the space of kernels

We can

Use backpropagation to search over the entire space of shift invariant kernels!

Continuous kernel learning

A two layer neural network whose hidden layer consists of cosine activation functions

$$y = \operatorname{sgn}\left(w_0 + \sum_{k=1}^{n} w_k \cos\left(\mathbf{u}_k^T \mathbf{x} + d_k\right)\right)$$

THE UNIVERSITY OF UTAH®

A new method for multiple kernel learning

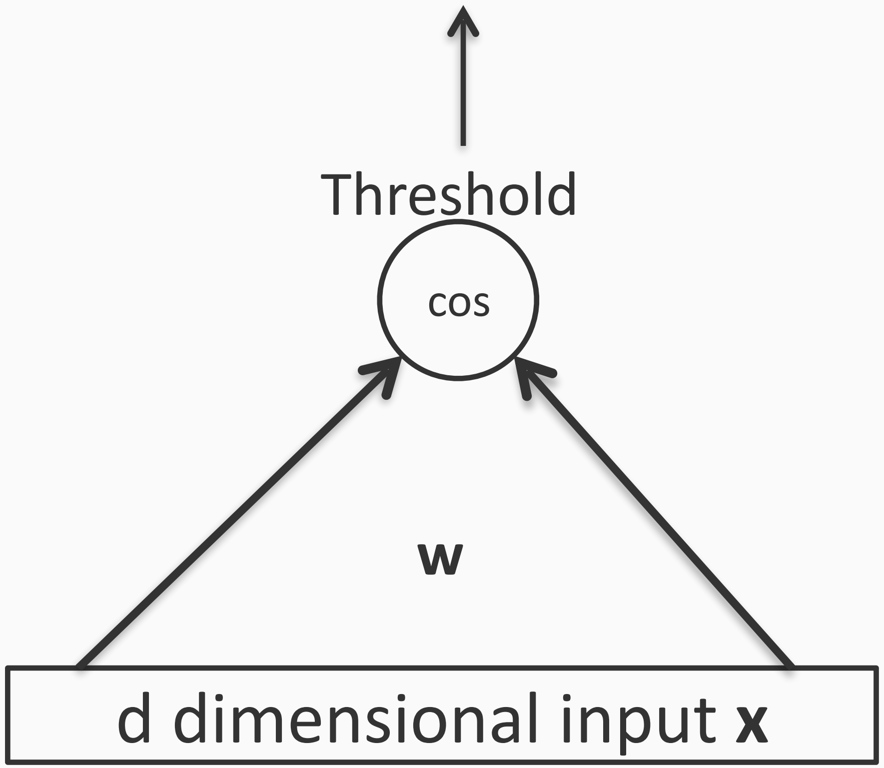
- Kernel learning: Learn a kernel from data

- **Multiple kernel learning** [Lanckriet et al 2004]: Assume that the kernel we want to learn is a linear combination of a set of "base" kernels
  - Often slow to train and predict, especially on larger datasets

- **Continuous kernel learning**: Using neural networks for kernel learning

What is the capacity of a single cosine unit?

VC dimension of a single cosine unit



General case: VC dimension is infinite

If we limit the norm of the weights to R, then

$$VC = \Theta\left(\max\left(d \log R, d + 1\right)\right)$$

Capacity control

VC dimension results suggests two different methods for regularization

1. Consider only **w**’s whose norm is less than R

2. Minimize the norm of **w**

Motivates L2 regularization for cosine neurons

What we have so far

- Two layer neural network with cosine hidden layers
- Training this network is searches over the space of shift invariant kernels
  - A new approach for multiple kernel learning: **Continuous kernel Learning**
  - Don’t know what kernel we will get at the end
- L2 regularization is a reasonable idea

Experiments: Is this better than other multiple kernel learning methods?

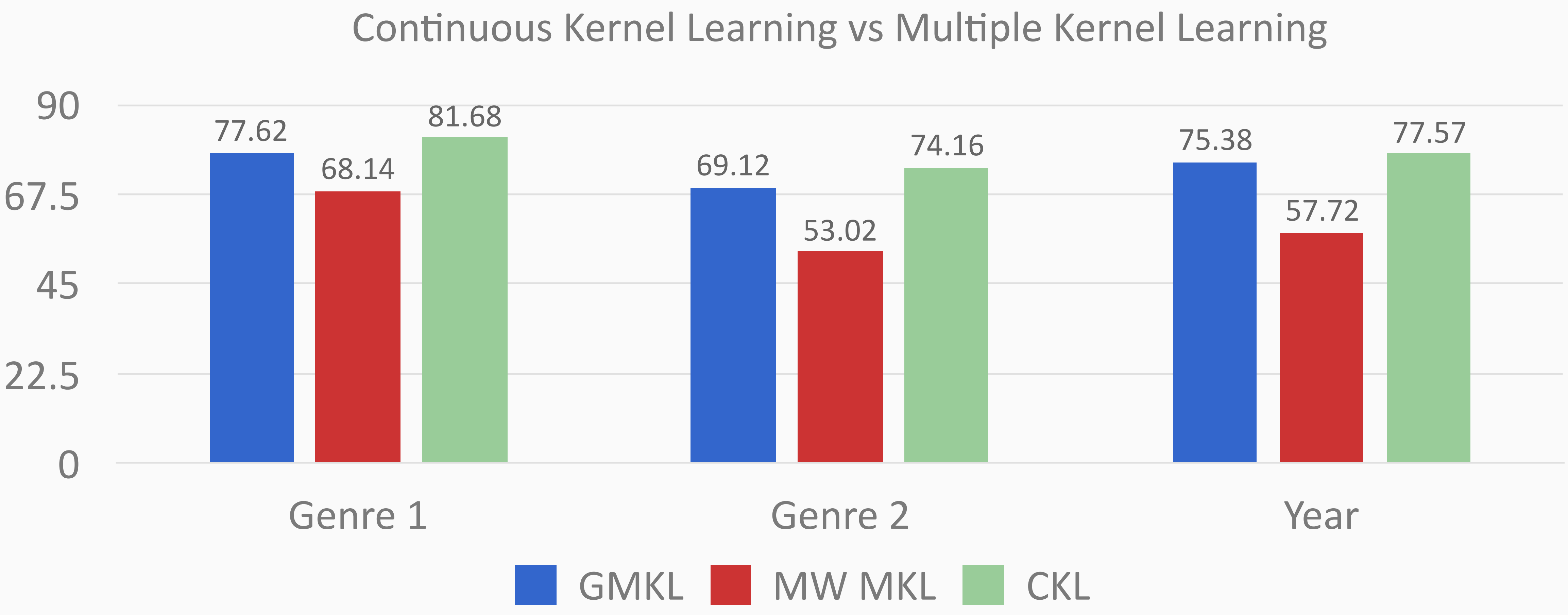
Million Song Dataset: The Setup

- One million songs
  - Audio features + metadata

- Three classification tasks (all binary)
  - Genre 1: *classic pop and rock* vs *folk*
  - Genre 2: *classic pop and rock* vs *all other genres*
  - Year: Was the song made before 2000 or after?

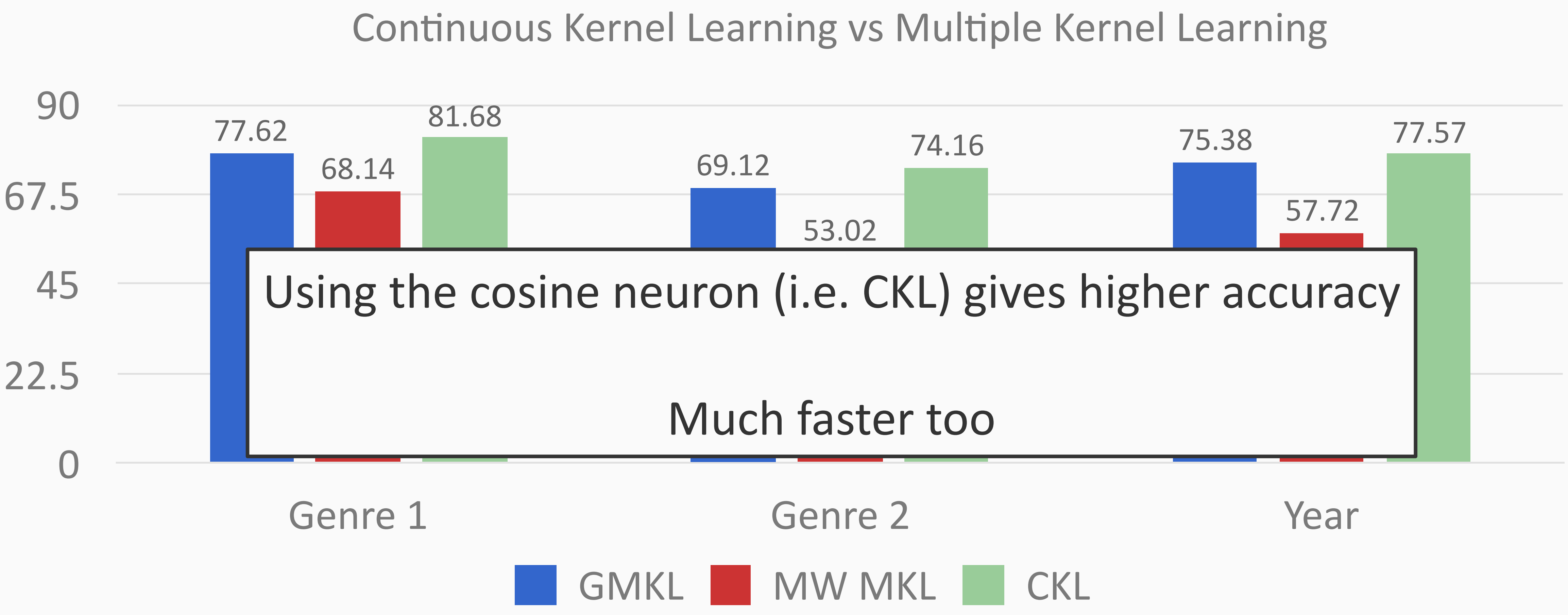
- Two multiple kernel learning baselines
  - SPG GMKL: Jain et al 2012
  - MW MKL: Moeller et al 2014

Million Song Dataset: Results



THE UNIVERSITY OF UTAH®

Million Song Dataset: Results



Using the cosine neuron (i.e. CKL) gives higher accuracy

Much faster too

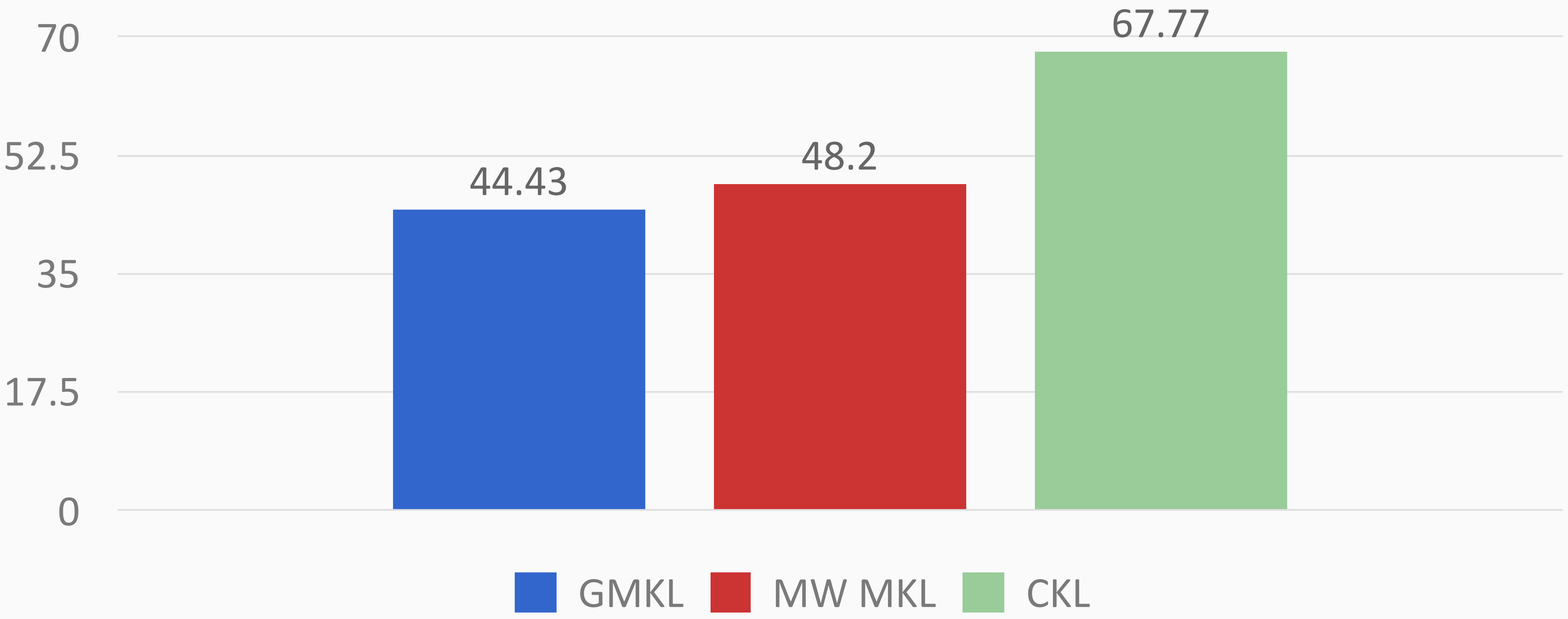
We can add more layers

If the **z**'s are only a representation, then why not a more complex classifier on top of it?

CIFAR-10:
- 32x32 images
- 10 classes
  - airplane, automobile, bird, cat, deer, dog, horse, frog, ship, truck

Preliminary experiments with a small convolutional network

Multiple kernel learning for CIFAR 10



Continuous Kernel Learning: Summary

- Shift-invariant kernels = probability distributions = cosine neural networks

- Bochner’s theorem does the heavy lifting

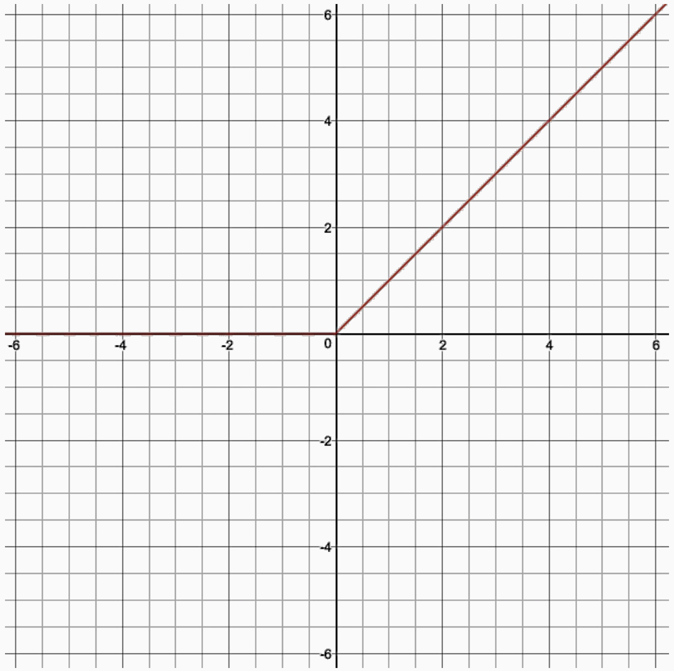
- Learning the cosine neural network with backpropagation searches over the space of kernels
  - Kernel learning via neural networks

Continuous Kernel Learning: Looking ahead

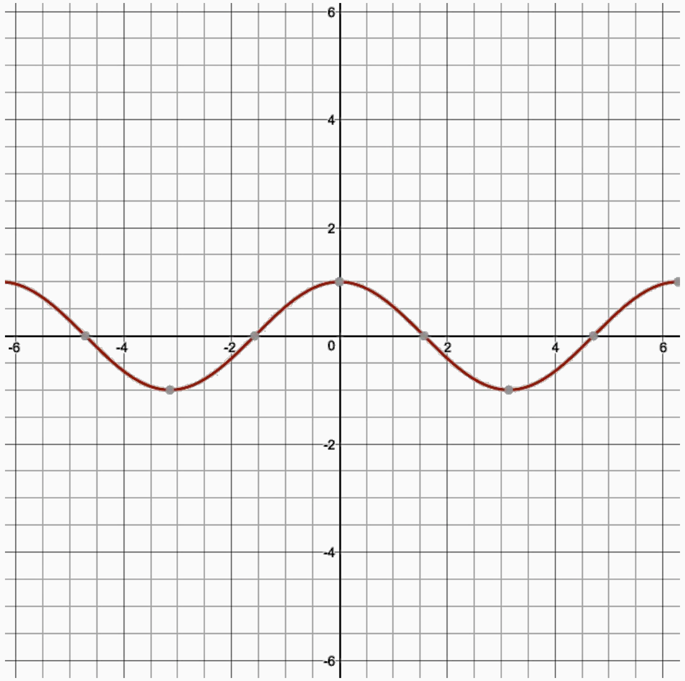
- Interplay of number of hidden units, regularization and sample complexity?
- Multiple layers?
- Empirical question: Applying this architecture to new problems

This talk: A tale of two activation functions

Rectified Linear Units



Cosine Neurons



Closing thoughts

- The gap between practice and theory
  - Empirically, neural networks work. But many choices…
  - Can we theoretically motivate some of these choices?

- Today's talk: Analyzing two activation functions
  - **Rectifier networks**: Two layer rectifier networks punch above their weight when it comes to expressivity
  - **Cosine neurons and continuous kernel learning**: Learning a cosine neural network to search over the space of shift invariant kernels

Closing thoughts

- The gap between practice and theory
  - Empirically, neural networks work. But many choices…
  - Can we theoretically motivate some of these choices?

- **Today's talk**: Analyzing two activation functions
  - **Rectifier networks**: Two layer rectifier networks punch above their weight when it comes to expressivity
  - **Cosine neurons and continuous kernel learning**: Learning a cosine neural network to search over the space of shift invariant kernels

Questions?