Towards a Deeper Understanding of Neural Networks

Zachary Charles, Alisha Zachariah

University of Wisconsin-Madison

rzachariah@math.wisc.edu
zcharles@math.wisc.edu

November 22, 2016

Motivation

Important questions about neural networks:

1. What functions can be expressed by neural networks?
2. When does low empirical risk imply low expected risk?
3. Why do algorithms such as SGD often find good weights?

Motivation

Important questions about neural networks:

1. What functions can be expressed by neural networks?
2. When does low empirical risk imply low expected risk?
3. Why do algorithms such as SGD often find good weights?

Focus: Why can we find good weights for NNs?

Paper Overview

- A NN learns a higher-dimensional representation of your data.
- How does that representation change with the weights?

Paper Overview

- A NN learns a higher-dimensional representation of your data.
- How does that representation change with the weights?

"Theorem": If our neural network has:

- enough redundancy
- random weights (no training),

then with high probability the neural network will encode "the same" representation of your data.

So what?

- Gives evidence for random initialization of weights.
- No need to train the hidden layer weights!
- Reduces neural network training to a *convex* problem.

So what?

- Gives evidence for random initialization of weights.
- No need to train the hidden layer weights!
- Reduces neural network training to a *convex* problem.

Where's the rub?

So what?

- Gives evidence for random initialization of weights.
- No need to train the hidden layer weights!
- Reduces neural network training to a *convex* problem.

Where's the rub?

- This guarantee requires a lot of redundant neurons.

So what?

- Gives evidence for random initialization of weights.
- No need to train the hidden layer weights!
- Reduces neural network training to a *convex* problem.

Where's the rub?

- This guarantee requires a lot of redundant neurons.

Still...

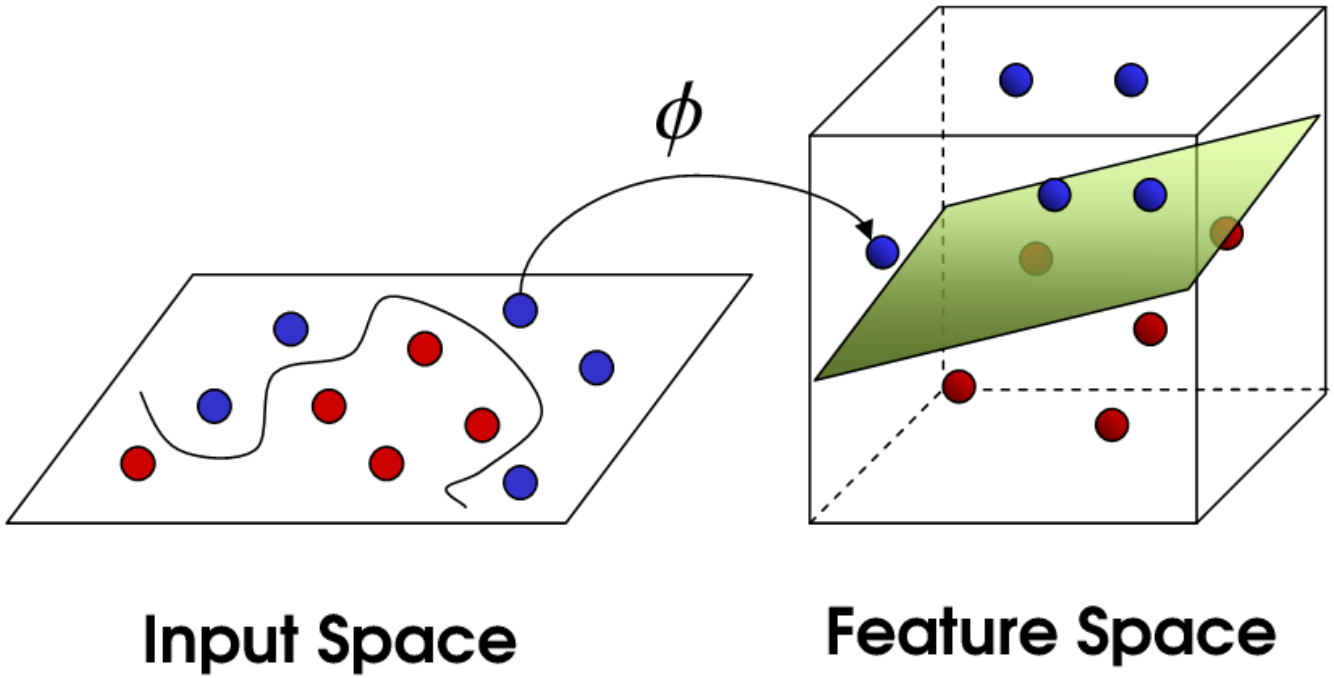
- Empirical evidence supports some of these claims.
- Better theoretical constants in the works?

Data representation

Problem: Classifiers don't always work on a given data set.

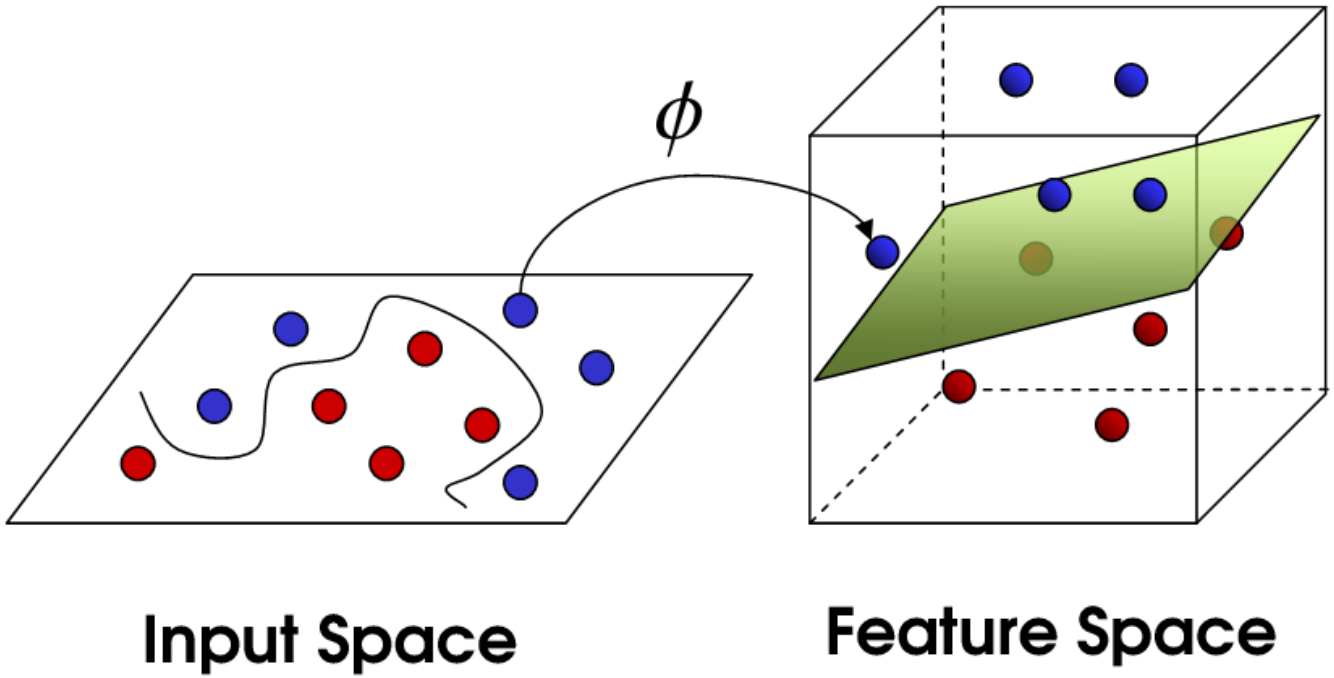
Data representation

Problem: Classifiers don't always work on a given data set.
Solution: Embed data in to higher dimensions.



Data representation

Problem: Classifiers don't always work on a given data set.
Solution: Embed data in to higher dimensions.



Want a map $\phi : X \to \mathcal{H}$, where $\mathcal{H}$ is a "nice" higher dimensional space.
This gives us a more rich representation of our data.

Kernels

To do machine learning on $\mathcal{H}$, we want a similarity measure.

- Mathematically, we want $\mathcal{H}$ with inner product $\langle x, y \rangle_{\mathcal{H}}$

For $x, y \in X$, define the **kernel** of $\mathcal{H}$ by:

$$K(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$$

Kernels

To do machine learning on $\mathcal{H}$, we want a similarity measure.

- Mathematically, we want $\mathcal{H}$ with inner product $\langle x, y \rangle_{\mathcal{H}}$

For $x, y \in X$, define the **kernel** of $\mathcal{H}$ by:

$$K(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$$

It works both ways! Given a kernel K (easy to compute), there is some associated $\phi : X \to \mathcal{H}$ (hard to compute).

Kernels

To do machine learning on $\mathcal{H}$, we want a similarity measure.

- Mathematically, we want $\mathcal{H}$ with inner product $\langle x, y \rangle_{\mathcal{H}}$

For $x, y \in X$, define the **kernel** of $\mathcal{H}$ by:

$$K(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$$

It works both ways! Given a kernel K (easy to compute), there is some associated $\phi : X \to \mathcal{H}$ (hard to compute).
Examples:

$$K(x, y) = \langle x, y \rangle \leftrightarrow \phi(x) = x$$
$$K(x, y) = \langle x, y \rangle^p \leftrightarrow \phi(x) = (1, x, x^2, \ldots, x^p)$$
$$K(x, y) = \exp(-||x - y||) \leftrightarrow \phi(x) = e^{-x^2}(1, x, x^2, \ldots)$$

Kernels

To do machine learning on $\mathcal{H}$, we want a similarity measure.

- Mathematically, we want $\mathcal{H}$ with inner product $\langle x, y \rangle_{\mathcal{H}}$

For $x, y \in X$, define the **kernel** of $\mathcal{H}$ by:

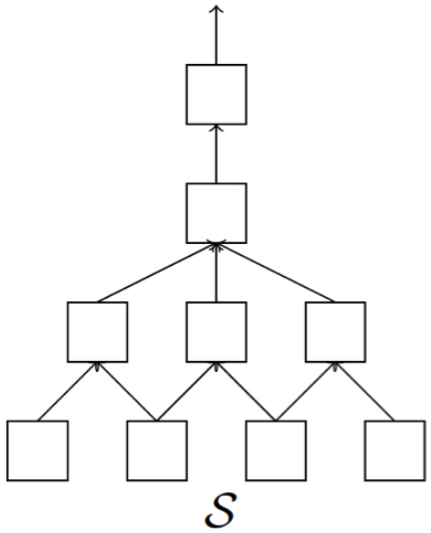
$$K(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$$

It works both ways! Given a kernel K (easy to compute), there is some associated $\phi : X \to \mathcal{H}$ (hard to compute).
Examples:

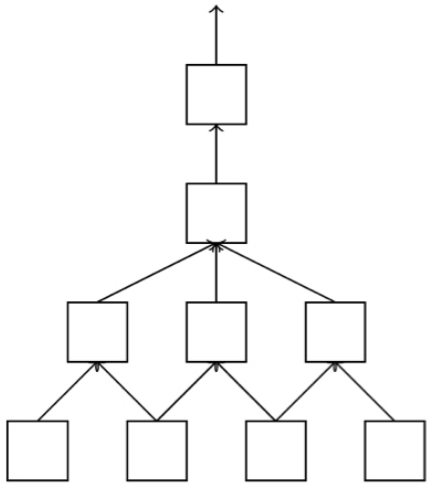
$$K(x, y) = \langle x, y \rangle \leftrightarrow \phi(x) = x$$
$$K(x, y) = \langle x, y \rangle^p \leftrightarrow \phi(x) = (1, x, x^2, \ldots, x^p)$$
$$K(x, y) = \exp(-||x - y||) \leftrightarrow \phi(x) = e^{-x^2}(1, x, x^2, \ldots)$$

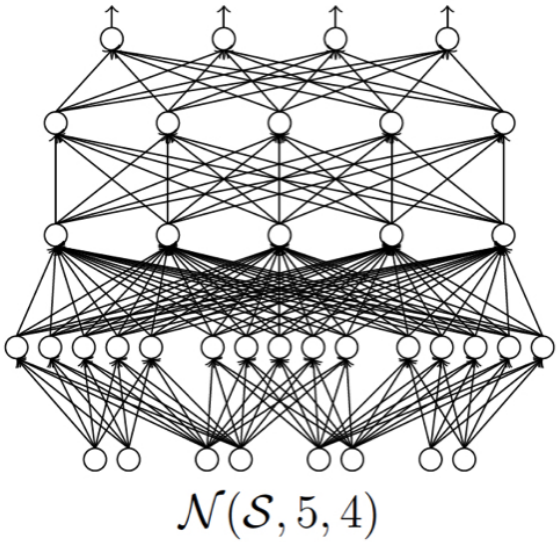
Bottom line: Kernels correspond to representations of your data.

The Computational Skeleton:

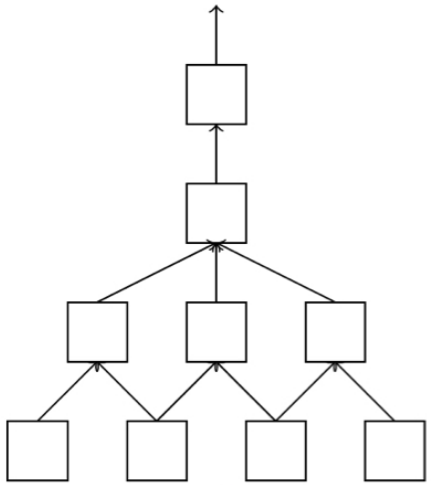


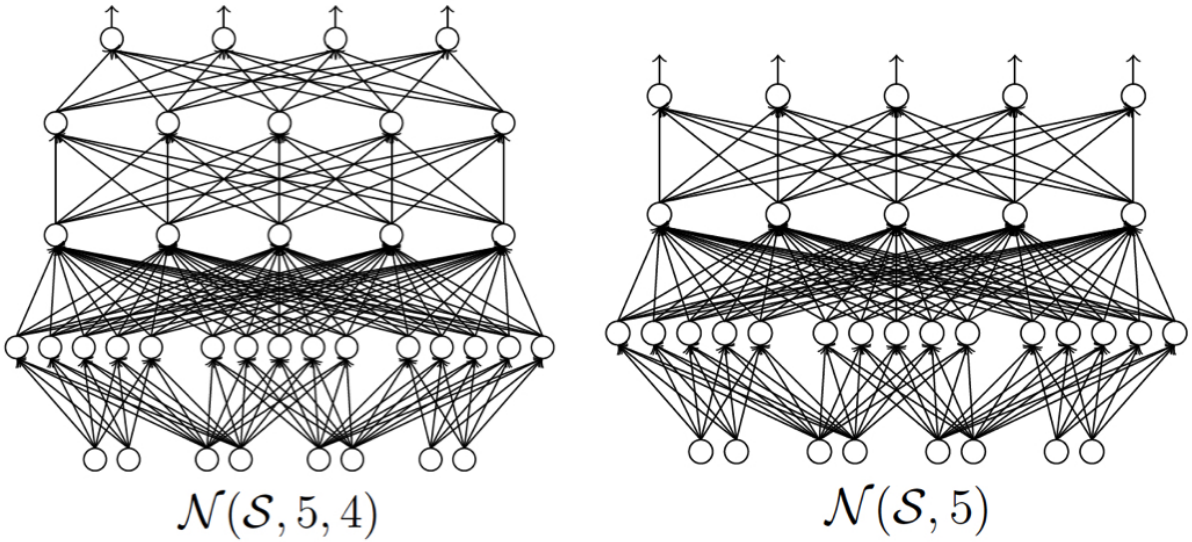
The Computational Skeleton:





The Computational Skeleton:

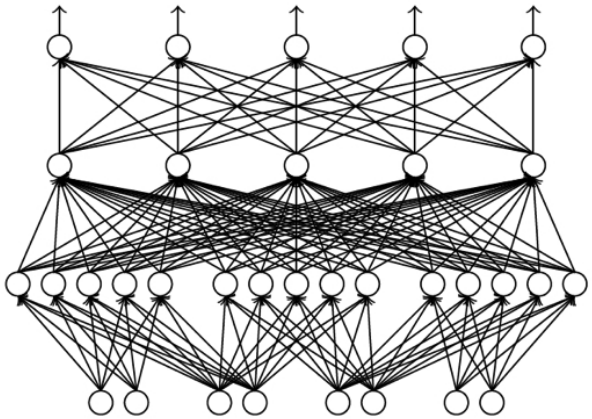




Computational Skeletons to Kernels:

w - given weights on edges of $\mathcal{N}(S, r)$

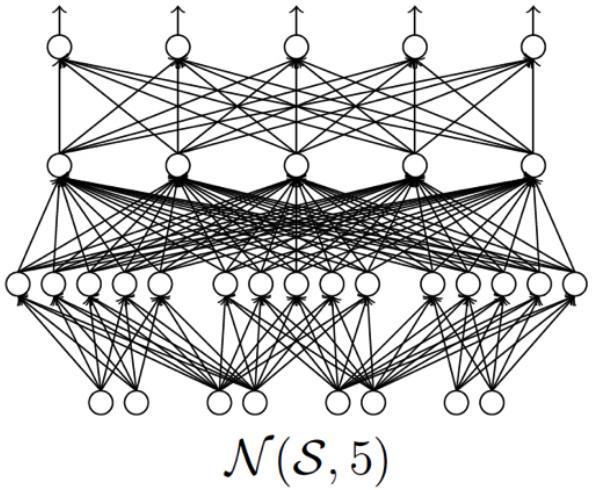
v - output node in representation



Computational Skeletons to Kernels:

w - given weights on edges of $\mathcal{N}(S, r)$

v - output node in representation



Empirical Kernel κ_w:

$$\Psi_w(x) \approx \text{ vector of (normalized) outputs indexed by } v$$

$$\kappa_w(x, x') = \langle \Psi(x), \Psi(x') \rangle$$

"Theoretical" Kernel:

Compositional Kernel κ_S:

Inductively define a kernel κ_v for each node v as follows:

- For an input node v define $\kappa_v(x, y) = \langle x, y \rangle$.
- For a non-input node v define

$$\kappa_v = \hat{\sigma}_v \left(\frac{\sum_{u \in \text{in}(v)} \kappa_u(x, y)}{|\text{in}(v)|} \right)$$

Pretend there is a single output node o, $\kappa_S = \kappa_o$

"Theoretical" Kernel:

Compositional Kernel κ_S:

Inductively define a kernel κ_v for each node v as follows:

- For an input node v define $\kappa_v(x, y) = \langle x, y \rangle$.
- For a non-input node v define

$$\kappa_v = \hat{\sigma}_v \left(\frac{\sum_{u \in \text{in}(v)} \kappa_u(x, y)}{|\text{in}(v)|} \right)$$

Pretend there is a single output node o, $\kappa_S = \kappa_o$

Dual Activation $\hat{\sigma}$:

Let $\Sigma_\rho = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$. Given an activation σ, the dual activation $\hat{\sigma} : [-1, 1] \to \mathbb{R}$ is defined as

$$\hat{\sigma}(\rho) = \mathop{\mathbb{E}}_{X, Y \sim \mathcal{N}(0, \Sigma_\rho)} \sigma(X)\sigma(Y)$$

Main Theorem:

Theorem:
ReLU activations, a random initialization of weights $w \sim \mathcal{N}(0, 1)$.
If

$$r \geq \frac{\text{depth}^2(S) \log(|S|/\delta)}{\epsilon^2}$$

Then for all x, x', with probability at least $1 - \delta$,

$$|\kappa_w(x, x') - \kappa_S(x, x')| \leq \epsilon$$

"Proof":

- An activation σ is (α, β, γ)-**decent** if the following holds:
  - The dual activation $\hat{\sigma}$ is β - Lipschitz (in a specific sense)
  - Concentration : for X_i, Y_i $i = 1, \ldots r \sim \mathcal{N}(0, \Sigma_\rho)$

$$\Pr\left(\left|\frac{\sum_i \sigma(X_i)\sigma(Y_i)}{r} - \hat{\sigma}(\rho)\right| \geq \epsilon\right) \leq 2\exp\left(-\frac{r\epsilon^2}{2C^4}\right)$$

"Proof":

- An activation σ is (α, β, γ)-**decent** if the following holds:
  - The dual activation $\hat{\sigma}$ is β - Lipschitz (in a specific sense)
  - Concentration : for X_i, Y_i $i = 1, \ldots r \sim \mathcal{N}(0, \Sigma_\rho)$

$$\Pr\left(\left| \frac{\sum_i \sigma(X_i)\sigma(Y_i)}{r} - \hat{\sigma}(\rho) \right| \geq \epsilon \right) \leq 2 \exp\left(- \frac{r\epsilon^2}{2C^4} \right)$$

- **Lemma :** ReLU is decent.
  - Measure concentration property from standard concentration bounds for sub-exponential random variables.
  - $\hat{\sigma}$ is Lipschitz - work.

"Proof":

- An activation σ is (α, β, γ)-**decent** if the following holds:
  - The dual activation $\hat{\sigma}$ is β - Lipschitz (in a specific sense)
  - Concentration : for X_i, Y_i $i = 1, \ldots r \sim \mathcal{N}(0, \Sigma_\rho)$

$$\Pr\left(\left|\frac{\sum_i \sigma(X_i)\sigma(Y_i)}{r} - \hat{\sigma}(\rho)\right| \geq \epsilon\right) \leq 2\exp\left(-\frac{r\epsilon^2}{2C^4}\right)$$

- **Lemma :** ReLU is decent.
  - Measure concentration property from standard concentration bounds for sub-exponential random variables.
  - $\hat{\sigma}$ is Lipschitz - work.
- **Theorem:** For (α, β, γ)-decent activations, random weights w for

$$r \geq \frac{2\alpha^2(\sum_{i=1}^{\text{depth}} \beta_i)^2 \log(|S|/\delta)}{\epsilon^2}$$

Moral:

- Supports empirical evidence in favour of random initialization of weights.

Moral:

- Supports empirical evidence in favour of random initialization of weights.
- Supports empirical evidence that contribution of optimizing the representation layers is relatively small – just train the last layer!
  - CIFAR-10, STL-10, MNIST and MONO datasets
  - Saxe et al., 2011, *On random weights and unsupervised feature learning*
  - Metrics induced by the initial and fully trained representations are not substantially different
  - Giryes et al. demonstrated for the MNIST and CIFAR-10 datasets

Moral:

- Supports empirical evidence in favour of random initialization of weights.
- Supports empirical evidence that contribution of optimizing the representation layers is relatively small – just train the last layer!
  - CIFAR-10, STL-10, MNIST and MONO datasets
  - Saxe et al., 2011, *On random weights and unsupervised feature learning*
  - Metrics induced by the initial and fully trained representations are not substantially different
  - Giryes et al. demonstrated for the MNIST and CIFAR-10 datasets
- Usefulness of ReLU – even for quite deep networks with ReLU activations, random initialization approximates the corresponding kernel.