Mathematics of Deep Learning
Lecture 2: Expressive/Approximation Power of Neural Nets

Prof. Predrag R. Jelenković
Time: Wednesday 4:10-6:40pm

Dept. of Electrical Engineering
Columbia University , NY 10027, USA
Office: 812 Schapiro Research Bldg.
Phone: (212) 854-8174
Email: predrag@ee.columbia.edu
URL: http://www.ee.columbia.edu/∼predrag

Last Lecture: General Learning Framework

- Supervised learning: there is an input-output relationship

$$\boxed{Y = f(X)}$$

  - f - unknown
  - Training data (observations): $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
  - Objectives:
    - Learn/estimate $\hat{f}$ from training data
    - Inference/prediction: Use $\hat{f}$ to predict outcomes on unseen/new data
  - Two problems:
    - Regression: Y is quantitative
    - Classification: Y is categorical
- Unsupervised learning: Just X, no output Y (no labels)
  Typical problems:
  - Distribution estimation: learn the distribution/density, $p(x)$
    - Generative modeling: first estimate the distribution, then use it for other learning problems, e.g., QDA/LDA
  - Dimensionality reduction, e.g., PCA
  - Clustering and, in general, any data mining, e.g., data association, etc.

Last Lecture: Curse of Dimensionality

Why is it hard to estimate a function in high dimensions?

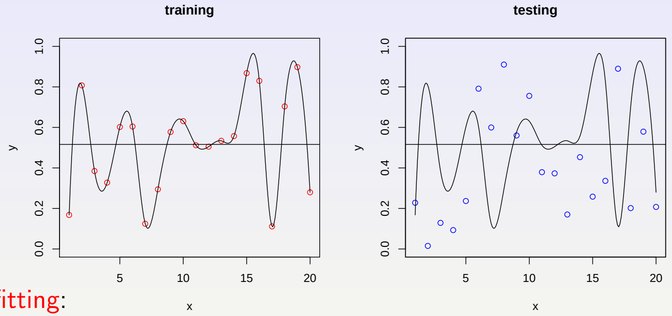
- How do we estimate a density of one dimensional X on $[0, 1]$?
  Question: Say, we want to learn $p(x) : [0, 1] \to \mathbb{R}$
  - Solution: split $[0, 1]$ in 100 bins of size $\epsilon = 0.01$, get about 1000 samples of X and plot a histogram
    This should be a pretty good estimate of $p(x)$
- Suppose X is supported on 100 dimensional cube $[0, 1]^{100}$
  Learn density $p(x) : [0, 1]^{100} \to \mathbb{R}$
  - The preceding solution: splitting $[0, 1]^{100}$ in bins of size 0.01^{100} would require more than

    10^{200} samples (!)

    This is usually referred to as curse of dimensionality
- Many real world problems are high dimensional
  Images $> 10^6$ dimensions; gene expression data $> 10,000$
- Only hope: existence of low dimensional structure

Overfitting Problem

One can fit infinitely many functions through a finite set of points, but



Overfitting:

- Low training error does not imply low testing error
- More complicated models not always better
  - Less interpretability
  - More difficult to train
- Mystery: Deep is learning highly flexible, possibly millions of parameters, but usually doesn't overfit.
- **John von Neumann elephant quote**: "With four parameters I can fit an elephant, and with five I can make him wiggle his trunk."

Last Lecture: General Supervised Learning Setup

Supervised learning problem can be formulated as (say, $x_i \in \mathbb{R}^p, y_i \in \mathbb{R}^m$)

$$\hat{f}^* = \arg\min_{\hat{f} \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \hat{f}(x_i)) + \lambda R(\hat{f}) \tag{1}$$

- $\ell : \mathbb{R}^{p+m} \to \mathbb{R}$ - loss function, and empirical risk/loss is defined as $\boldsymbol{x} = (x_1, \ldots, x_p), \boldsymbol{y} = (y_1, \ldots, y_m)$
$$\bar{L}(\boldsymbol{x}, \boldsymbol{y}) \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \hat{f}(x_i))$$
Hence, Equation (2) is called Empirical Risk Minimization (ERM)
- $\mathcal{H}$ - Hypothesis class (class of approximation functions)
Desirable properties of $\mathcal{H}$:
  - Rich/versatile, yields accurate predictions, easy to train (e.g., problem (2) is convex), interpretable/simple, etc.
- $\lambda R(\hat{f}) \in \mathbb{R}^+$ - regularizer/penalty, shrinkage term
  - $\lambda R(\hat{f})$ - shrinking $\mathcal{H}$: if $\lambda_2 > \lambda_1 \to \mathcal{H}_{\lambda_2} \subset \mathcal{H}_{\lambda_1}$
  - Should prevent overfitting

Linear Examples: Regression and Classification

Regularized Linear Regression (say $y \in \mathbb{R}$)

- $\mathcal{H}$: set off all p-dimensional linear functions
  $\hat{f}(x) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$
- Quadratic loss: $\ell(y_i, \hat{f}(x_i)) = (y_i - \hat{f}(x_i))^2$
- Typical penalties
  - Ridge - l_2 norm: $R(\hat{f}) =< \hat{f}, \hat{f} >= \|f\|^2 = \sum \beta_j^2$
  - LASSO: l_1 norm (basis pursuit): $R(\hat{f}) = \sum |\beta_j|$

Support Vector Classifier: Separate two classes by a hyperplane, $y_i \in \{-1, 1\}$

$$\min_{\beta_0, \boldsymbol{\beta}} \sum_{i=1}^{n} \max\left[0, 1 - y_i(\beta_0 + x_{i1}\beta_1 + \cdots + x_{ip}\beta_p)\right] + \lambda \sum_{j=0}^{p} \beta_j^2$$

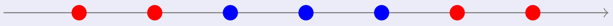
$\ell(x_i, y_i) = \max\left[0, 1 - y_i(\beta_0 + x_{i1}\beta_1 + \cdots + x_{ip}\beta_p)\right]$ is called **hinge loss**

- What if the problem is not linear?
  This is most likely the case.

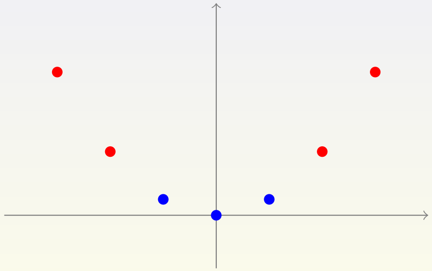
Nonlinear Functions: Transform/Expand Features

Example: Classifying red and blue point on a real line

- Can't be separated by a single point (hyperplane)



- Nonlinear feature transformation/expansion: $x \to (x, x^2)$
- Now, the points are separable by a single hyperplane (!)



- Feature engineering: $x \to \phi(x)$
- Well-developed theory: Reproducing Kernel Hilbert Spaces (RKHS)
- Problem: How do we find feature functions/kernels?

Where Does Deep Learning Fits in This Framework?

- Rich $\mathcal{H}$: Provides a versatile parametric class of functions
  - Universal function approximation: depth helps improves expressiveness
  - However, it is difficult to train: non-convex optimization
  - Often produces accurate predictions in practice, but difficult to understand and interpret
- Automatic feature extraction
  - Traditional Feature Engineering approach: expert constructs feature mapping $\phi : \mathcal{X} \to \Phi$. Then, apply machine learning to find a linear predictor on $\phi(\mathbf{x})$.
  - "Deep learning" approach: neurons in hidden layers can be thought of as features that are being learned automatically from the data
  - Shallow neurons corresponds to low level features, while deep neurons correspond to high level features

Parametric Supervised Learning

- $\mathcal{H}$ is a parametric class of functions $f(w,x), w \in \mathcal{W}, w = (w_1, \ldots, w_k)$
  - Examples: polynomials, or other linear combinations of basis, neural networks
- Finding $f \in \mathcal{H}$ is equivalent to finding $w \in \mathcal{W}$

Hence, our general supervised learning problem can be formulated in terms of w as (say, $x_i \in \mathbb{R}^p, y_i \in \mathbb{R}^m$,$\ell : \mathbb{R}^{p+m} \to \mathbb{R}$ - loss function)

$$\hat{w} = \arg\min_{w \in \mathcal{W}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(w, x_i)) + \lambda R(w) \tag{2}$$

How do we solve the preceding problem?

- When the problem is convex and we are lucky, we can find an explicit $\hat{w}$ by solving (e.g., generalized (Kernel) ridge regression)

$$\frac{\partial}{\partial w_i} \left(\frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(w, x_i)) + \lambda R(w) \right) = 0, \qquad i = 1, \ldots, k.$$

Parametric Supervised Learning

Let us denote the objective function

$$F(w) := \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(w, x_i)) + \lambda R(w)$$

In general, we use the following numerical algorithm:

1. Initialization: Pick an initial value w_0, possibly random
2. Iteration: Follow the path of steepest descent = direction of negative gradient.

For the preceding procedure to find a local minimum

- Avoid getting stuck on a flat surfaces, say flat saddle points.
- If F is convex, we can find a global minimum.

Properties of gradient, $\nabla f(x)$:

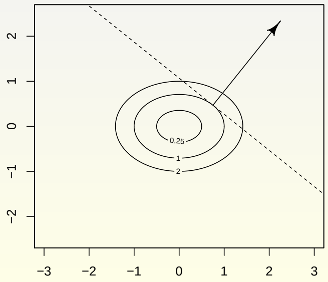
$\nabla f(x)$: direction of steepest ascent/max increase of $f(x)$

$-\nabla f(x)$: direction of steepest descent/max decrease of $f(x)$

$\nabla f(x)$ is perpendicular to level curves $f(x) = c$

Path of Gradient Descent

- $\mathbf{x}(t)$ - path of steepest gradient descent given initial condition $\mathbf{x}(0)$ is given by an ODE
$$\frac{d\mathbf{x}(t)}{dt} = -\eta \nabla f(\mathbf{x}(t)), \tag{3}$$
where η is the rate/speed of descent, a.k.a. learning rate. Note that $x'(t)$ is tangent to the curve $x(t)$, and thus parallel to $\nabla f(\mathbf{x}(t))$.
- Example: $f(\mathbf{x}) = ax_1^2 + bx_2^2, a, b > 0, \Rightarrow$
$\nabla f(\mathbf{x}) = (2ax_1, 2bx_2) \Rightarrow x_1'(t) = -2\eta a x_1(t), x_2'(t) = -2\eta b x_2(t)$
$$x_1(t) = x_1(0)e^{-2\eta at}, \qquad x_2(t) = x_2(0)e^{-2\eta bt}$$



Gradient Descent Algorithm

GD Algorithm is a discrete linear approximation to Equation (3)

$$\frac{\mathbf{x}(t+\Delta t)-\mathbf{x}(t)}{\Delta t} \approx \frac{d\mathbf{x}(t)}{dt} = -\eta \nabla f(\mathbf{x}(t))$$

or equivalently (with a small abuse of notation)

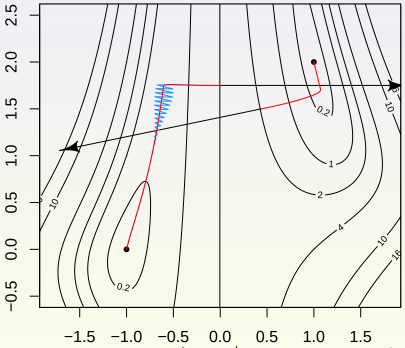
$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta_t \nabla f(\mathbf{x}^{(t)})$$

- Hence, after initialization at $\mathbf{x}^{(0)}$, the GD Algorithm follows the preceding iteration to a local minimum
- Stopping criterion (could be): $|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}| < \epsilon$

Adaptive GD (AdaGrad): modifies the learning rate η_t in each iteration t

More Interesting Landscape

- $f(x) = (x_1^2 - 1)^2 + (x_1^2 x_2 - x_1 - 1)^2$
- Gradient
$$\nabla f(x) = \begin{bmatrix} 4x_1(x_1^2 - 1) + 2(2x_1x_2 - 1)(x_1^2x_2 - x_1 - 1) \\ 2x_1^2(x_1^2x_2 - x_1 - 1) \end{bmatrix}$$
- Oscillations in "narrow valleys"



Motivation for momentum: remembers/averages previous Δx

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta_t \nabla f(\mathbf{x}^{(t)}) + \mu_t(\mathbf{x}^{(t)} - \mathbf{x}^{(t-1)})$$

Stochastic Gradient Descent

- Stochastic approximation of gradient descent
- Function (typically encountered in learning)

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

- Computationally expensive gradient for large n
- Approximation: pick a random subset $\mathcal{S} \in [1, n]$

$$\frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \nabla f_i(x)$$

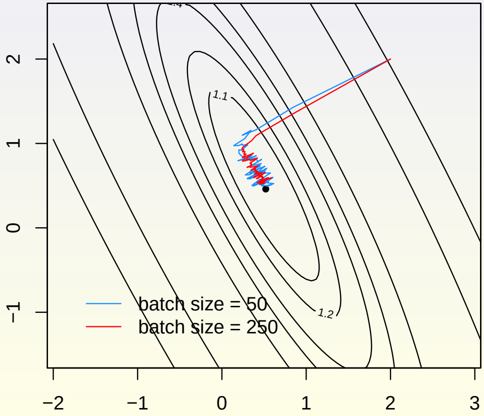
- $\mathcal{S}$ - called batch/mini-batch
- Example
  - n scalar data points $x_1, x_2, \cdots, x_n$
  - objective

$$\min_c \frac{1}{n} \sum_{i=1}^{n} (x_i - c)^2$$

SGD Example: Linear Regression

- Data: $(x_i, y_i)_{i=1}^n$, $n = 10^5$
- Loss function: $L(\beta, x, y) = \frac{1}{n}\sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2$
- Stochastic gradient

$$\nabla_\beta \hat{L}(\beta, x, y) = \frac{2}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\begin{bmatrix}(\beta_0 + \beta_1 x_i - y_i)\\ x_i(\beta_0 + \beta_1 x_i - y_i)\end{bmatrix}$$

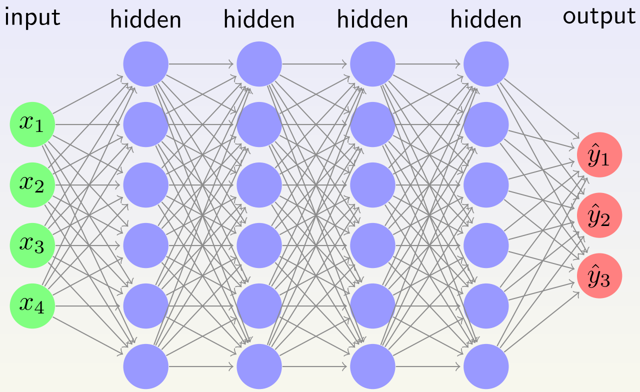


Math of Deep Learning: The Preliminary List of Topics

- **The preliminary list of topics** include the results on:
  - Expressive (approximation) power of neural networks
  - Depth separation results: Can deep neural networks of depth $(d+1)$ express functions much more efficiently in terms of the number of neurons and parameters compared to networks of depth d? What classes of functions can deep neural nets approximate well?
  - Connection between wide neural nets and Kernels: Neuro Tangent Kernels (NTK)
  - Global versus local optimality
  - Training and convergence properties of wide neural nets:
    - Is training converging to global min?
    - Is training converging far or close to the initial NTK?
  - Generalization error: Deep neural networks have a lot of parameters. How come they are not overfitting?
    - Basic generalization concepts from machine learning theory will be covered
  - Deep residual networks
  - Deep generative probabilistic models, e.g., deep Boltzmann machines (time permitting)
  - Etc.

Deep Neural Networks, a.k.a. Multilayer Perceptrons

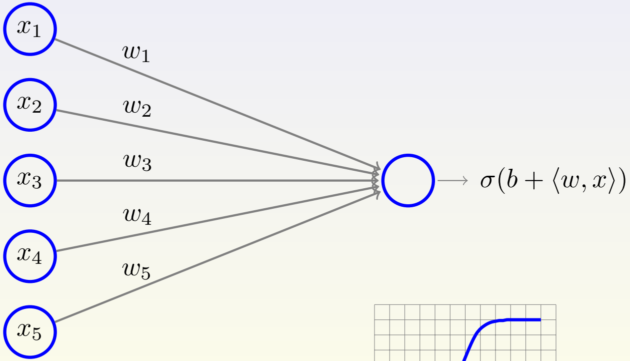
- Feed-forward network



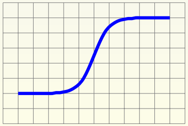
- Designed to mimic the function of neurons
- Blue nodes: activation functions/neurons
- Depth $=$ lengths of a longest path
- Deep network: depth ≥ 3
- Very successful in solving practical problems

A Single Artificial Neuron

- A single neuron function: $\mathbf{x} \mapsto \sigma(b + \langle \mathbf{w}, \mathbf{x} \rangle)$, where $\sigma : \mathbb{R} \to \mathbb{R}$ is called the activation function of the neuron. Inner/dot product: $\langle \mathbf{w}, \mathbf{x} \rangle = \sum x_i w_i$.
- More compact notation $\langle \tilde{\mathbf{w}}, \tilde{\mathbf{x}} \rangle$, where $\tilde{\mathbf{x}} = (1, \mathbf{x}), \tilde{\mathbf{w}} = (b, \mathbf{w})$



- E.g., σ is a sigmoidal function



Common Activation Functions/Perceptrons

- step function

$$\sigma(x) = 1_{\{x>0\}} \qquad \sigma'(x) = 0,\ x \neq 0$$

- logistic

$$\sigma(x) = \frac{1}{1+e^{-x}} \qquad \sigma'(x) = \sigma(x)(1-\sigma(x))$$

- rectified linear unit (ReLU)

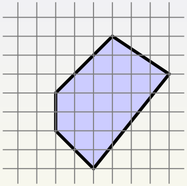
$$\sigma(x) = \max\{x, 0\} \qquad \sigma'(x) = 1_{\{x>0\}},\ x \neq 0$$

- soft-plus

$$\sigma(x) = \log(1+e^{x}) \qquad \sigma'(x) = \frac{1}{1+e^{-x}}$$

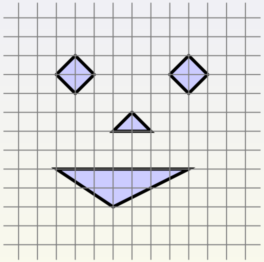
Example

- Single neuron is a binary half-space classifier: $\text{sign}(w \cdot x + b)$
- 2 layer networks can express intersection of halfspaces



Example

- 3 layer networks can express unions of intersection of halfspaces



How do we train neural networks?

- Neural nets: excellent hypothesis class, but difficult to train
- Main technique: Stochastic Gradient Descent (SGD)
- Not convex, no guarantees, can take a long time, but:
  - Often still works fine, finds a good solution
  - With some luck:)

Stochastic Gradient Descent (SGD) for Neural Networks

Common Training Ideas:

- Random initialization: rule of thumb, $w[u \to v] \sim U[-c, c]$ where $c = \sqrt{3/|\{(u', v) \in E\}|}$ (or small Gaussian instead of $U[-c, c]$)
- Update step with Nesterov's momentum: Initialize $\theta = 0$ and:

$$\theta_{t+1} = \mu_t \theta_t - \eta_t \tilde{\nabla} L(w_t + \mu_t \theta_t)$$
$$w_{t+1} = w_t + \theta_{t+1}$$

  where:
  μ_t is momentum parameter (e.g. $\mu_t = 0.9$ for all t)
  η_t is learning rate (e.g. $\eta_t = 0.01$ for all t)
  $\tilde{\nabla} L$ is an estimate of the gradient of L based on a small set of random examples (often called a "minibatch")
- Efficient gradient calculation: Backpropagation

Expressive/Approximation Power of Neural Nets

Question: What types of functions can be approximated with neural networks?

Two approaches to answering this question:

- Constructive approach: for a given function, we construct explicitly its neural network approximation.
  - This is usually more intuitive, but less general.
- Existence approach: we prove that for a given class of functions, there exists an accurate neural network approximation.
  - This gives us confidence that good approximations exist, but less informative.

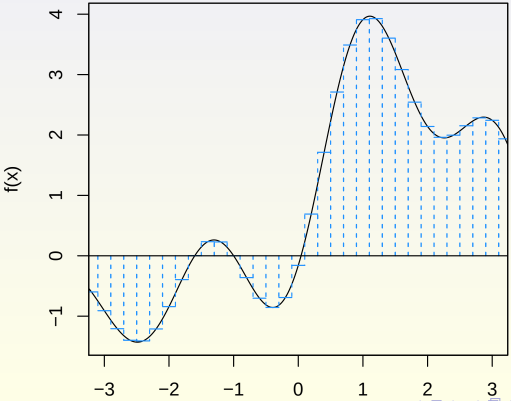
Expressive Power of Shallow Neural Nets

Questions: What type of real-valued functions can we approximate with NNs with one hidden layer? Such NN approximations can be written as:

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} a_i \sigma(\langle \boldsymbol{w}_i, \boldsymbol{x} \rangle + b_i), \quad \boldsymbol{x}, \boldsymbol{w} \in \mathbb{R}^d, a_i, b_i \in \mathbb{R}$$

Single variable example: explicit approximation with simple/step unctions

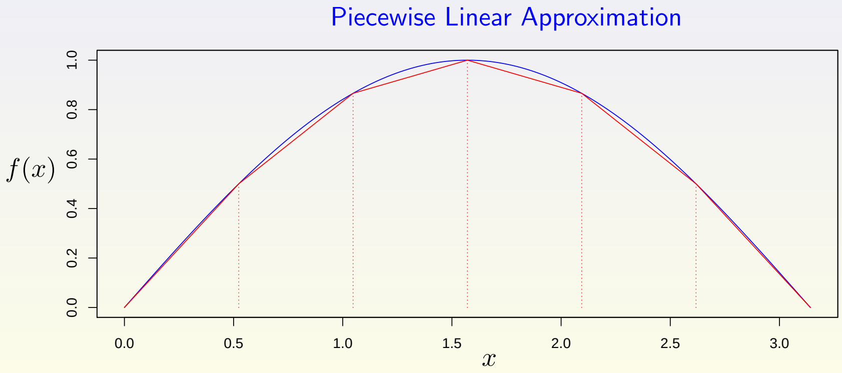
$$\sigma(x) = 1_{\{x>0\}}, \qquad \phi(x) = 1_{\{x>0\}} - 1_{\{x>1\}} \quad \text{(simple/pulse function)}$$



Expressive Power of Neural Nets

Piecewise Linear Approximation: ReLU - $\sigma(x) = \max(0, x)$
Example: $f(x) : [0, \pi] \to [0, 1]$

$$f(x) = \sin(x)$$



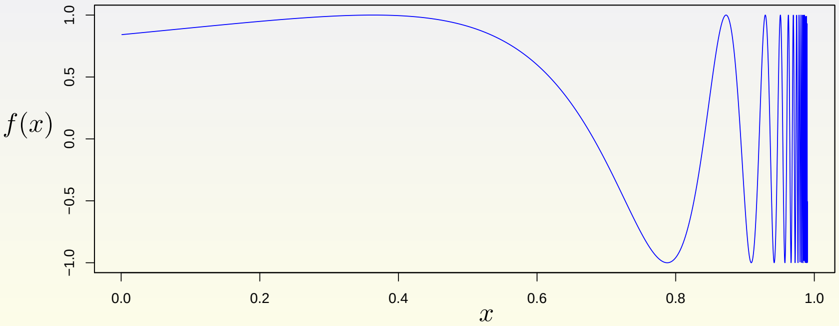
Example: Function That Is Difficult to Approximate

Consider function $f(x) : [0, 1) \to [-1, 1]$

$$f(x) = \sin\left(\frac{1}{1-x}\right)$$

is continuous and infinitely differentiable.

Difficult to Approximate: Why?



Function Regularity: Uniform Continuity

So, we need to impose some regularity on $f(x)$.
Definition Let $I \subset \mathbb{R}$ be an interval. Function $f(x) : I \to \mathbb{R}$ is uniformly continuous if for any $\epsilon > 0$, there exists $\delta > 0$, such that

$$|x_1 - x_2| < \delta \quad \Rightarrow |f(x_1) - f(x_2)| < \epsilon$$

Remarks

- If I is a closed interval, $[a, b]$, then

  ordinary continuity $\Leftrightarrow$ uniform continuity

- Hence, to prevent the problem from the preceding example, $f(x)$ needs to be defined on a closed interval $[0, 1]$ instead of $[0, 1)$.
- The preceding definition extends to $\mathbb{R}^k$ spaces with Euclidean norm, and in general metric spaces.
- Finite closed intervals generalize to compact sets.

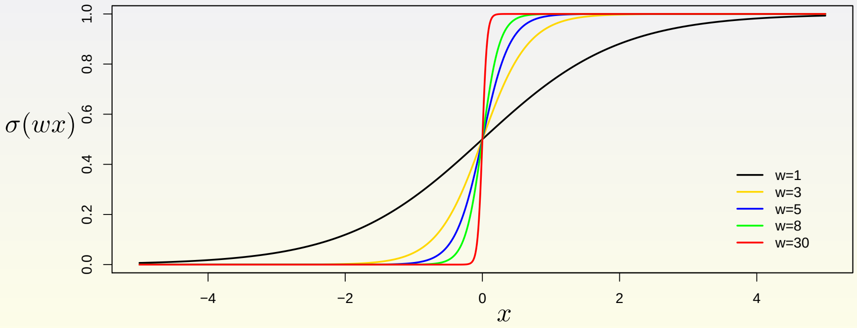
From Sigmoidal to Step Function

Starting with any sigmoidal functions, say

$$\sigma(x) = \frac{1}{1+e^{-x}},$$

we can approximate arbitrarily close a step function by using $\sigma(wx)$ and w large enough.

$\sigma(wx), w = 1, 3, 5, 8, 30$



Making Simple/Pules Function, a.k.a. Haar Wavelet

Haar scaling function (single pulse) is defined

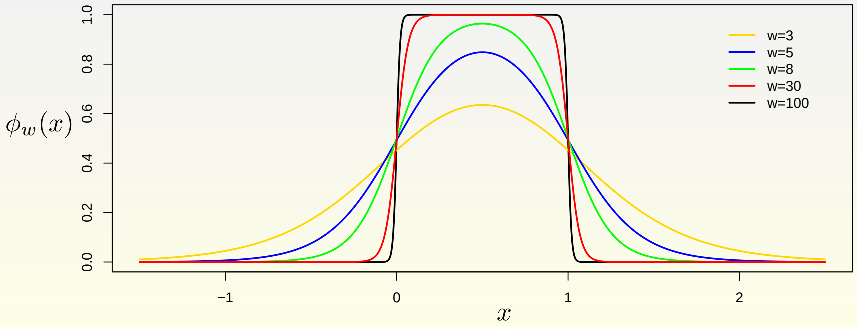
$$\phi(x) = 1_{\{x>0\}} - 1_{\{x-1>0\}}$$

Two sigmoidal functions, we can create a perfect pulse (Haar function)

$$\phi(x) \approx \phi_w(x) = \sigma(wx) - \sigma(w(x-1))$$

for w large enough (which can be used to make Haar wavelet basis)

$\phi_w(x), w = 3, 5, 8, 30, 100$



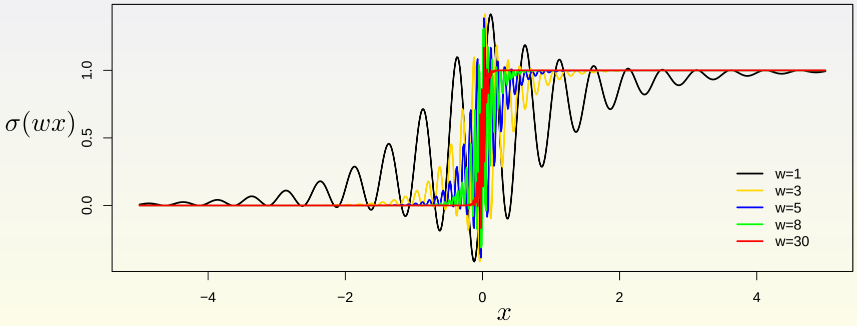
Non-monotonic Sigmoidal to Step Function

Do we need sigmoidal functions to be monotonic? For example

$$\sigma(x) = \frac{1}{1 + e^{-x}} + \sin(4\pi x)e^{-|x|}$$

Again, we can approximate arbitrarily close a step function by using $\sigma(wx)$ and w large enough.

$\sigma(wx), w = 1, 3, 5, 8, 30$



Continuity can be relaxed to $\sigma(x)$ being bounded with limtis at $\pm\infty$.

Universal Approximation Theorem in 1D

Definitions

- A function $\sigma(x) : \mathbb{R} \to \mathbb{R}$ is called sigmoidal if it is bounded and

$$\lim_{x\to-\infty} \sigma(x) = 0 \quad \text{and} \quad \lim_{x\to\infty} \sigma(x) = 1.$$

  (Equivalence to step function: $\sigma(wx) \approx 1_{\{x>0\}}$ for large w.)

- $C([a,b])$: space of continuous functions, $f(x) : [a,b] \to \mathbb{R}$ without loss of generality, we consider $C([0,1])$

Theorem (Constructive/Explicit Representation) Consider a sigmoidal function, σ and $f \in C([0,1])$. For every $\epsilon > 0$, there exist an integer n and $w > 0$ (depending on n), such that for $x \in [0,1]$

$$\hat{f}(x) \overset{\text{def}}{=} \sum_{k=1}^{n} (f(k/n) - f((k-1)/n))\sigma(w(x-k/n)) + f(0)\sigma(w(x+1/n)),$$

then

$$\sup_{0\le x\le 1} |\hat{f}(x) - f(x)| < \epsilon.$$

Universal Approximation Theorem in 1D: Proof

- Let $\hat{g}$ be a simple function approximation of f

$$\hat{g}(x) := \sum_{k=1}^{n} (f(k/n) - f((k-1)/n))1_{\{x-k/n>0\}} + f(0)1_{\{x+1/n>0\}}$$

- Since $f(x)$ is continuous on $[0,1]$, and thus uniformly continuous, we can chose n large enough, such that for any $x \in [0,1]$

$$|f(x) - \hat{g}(x)| \leq \frac{\epsilon}{2}$$

- To complete the proof, we need to bound the error between $\hat{f}(x)$ and $\hat{g}(x)$, i.e.,

$$|f(x) - \hat{f}(x)| \leq |f(x) - \hat{f}(x)| + |\hat{f}(x) - \hat{g}(x)| = \frac{\epsilon}{2} + |\hat{f}(x) - \hat{g}(x)|$$

- To this end, let $M := \max(1 + \sup_x \sigma(x), \sup_x f(x))$

Universal Approximation Theorem in 1D: Proof

We have two parameters to choose, n, w,

- n large enough to ensure

$$|f(k/n) - f((k-1)/n)| \leq \frac{\epsilon}{4M}$$

- w to make $\sigma(wx)$ arbitrarily close to a step function, i.e., we can choose w large enough such that for all $|x| \geq 1/n$

$$|\sigma(wx) - 1_{\{x>0\}}| \leq \frac{\epsilon}{4Mn}$$

  and around $x = 0$, since $\sigma(x)$ is bounded, for $|x| \leq 1/n$

$$|\sigma(wx) - 1_{\{x>0\}}| \leq 1 + \sup_x \sigma(x) \leq M$$

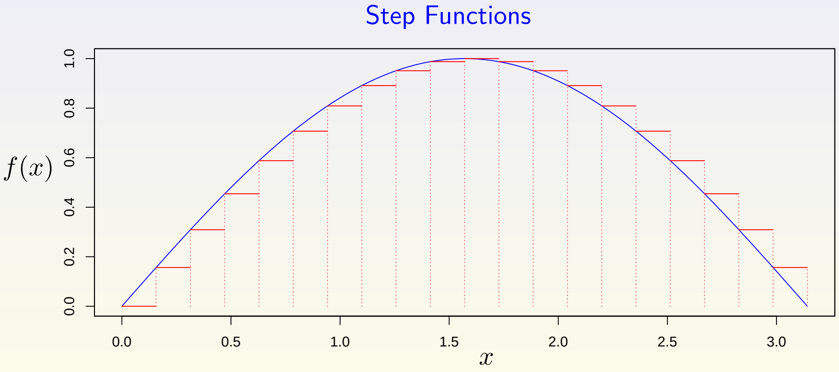
Hence, for $x \in [(j-1)/n, j/n]$,

$$\begin{aligned}|\hat{f}(x) - \hat{g}(x)| &\leq \sum_{k \neq j} |f(k/n) - f((k-1)/n)||\sigma(w(x-k/n)) - 1_{\{x-k/n>0\}}| \\ &+ |f(j/n) - f((j-1)/n)||\sigma(w(x-j/n)) - 1_{\{x-j/n>0\}}| \\ &+ |f(0)||\sigma(w(x+1/n)) - 1_{\{x+1/n>0\}}| \\ &\leq \frac{\epsilon}{4M}(n-1)\frac{\epsilon}{4Mn} + \frac{\epsilon}{4M}M + \frac{|f(0)|\epsilon}{4Mn} < \frac{\epsilon}{2}\end{aligned}$$

Sigmoidal Approximation Example

Consider function $f(x) : [0, \pi] \rightarrow [0, 1]$

$$f(x) = \sin(x)$$



Expressiveness of ReLU in 1D: Also Universal

- Recall, ReLU activation: $\sigma(x) = \max\{0, x\} =: x^+$
- To show that ReLU is universal, we can define a sigmoid $\sigma_1(x) = x^+ - (x-1)^+$ and use the preceding theorem
- Or we can do it directly using piece-wise linear approximation
- The following approximation can be made arbitrarily close to $f \in C([0,1])$, i.e., $|f(x) - \hat{f}(x)| < \epsilon$, for large enough n

$$\hat{f}(x) = \sum_{k=0}^{n-1} w_k \sigma(x - k/n) + f(0),$$

  where the slopes w_k are chosen such that $\hat{f}(k/n) = f(k/n)$, i.e.,

  - $w_0 = n(f(1/n) - f(0))$
  - w_1 is such that $f(2/n) = 2w_0/n + w_1/n + f(0)$, yielding

$$w_1 = n(f(2/n) - 2f(1/n) + f(0))$$

  - And so on, we inductively select w_k to satisfy

$$f\left(\frac{k+1}{n}\right) = \hat{f}\left(\frac{k+1}{n}\right) = w_0\frac{k+1}{n} + w_1\frac{k}{n} + \cdots + w_k\frac{1}{n} + f(0)$$

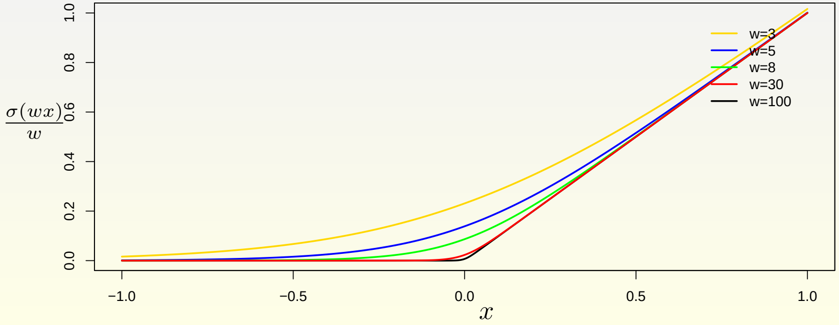
Functions Equivalent to ReLU

- Similarly to sigmoidal functions, through scaling, many other functions are equivalent to ReLU. For example, one can define ReLU-equivalent functions as: continuous with

$$\lim_{x\to-\infty} \sigma(x) = 0, \qquad \lim_{x\to-\infty} \frac{\sigma(x)}{x} = 1.$$

- Then, as $w \to \infty$, $$\frac{\sigma(wx)}{w} \to \max(0, x)$$

Scaling soft-plus $\sigma(x) = \log(1 + e^x)$ to ReLU

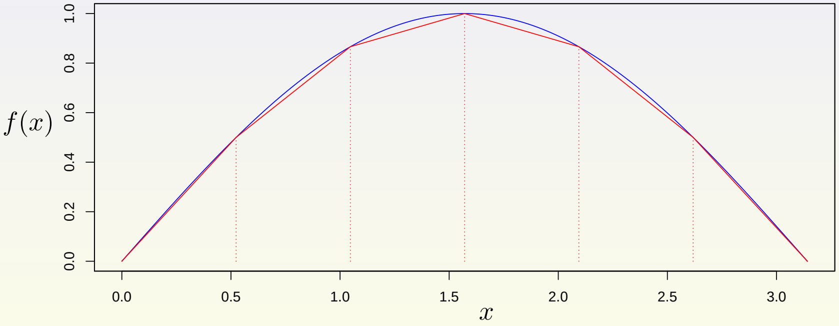


ReLU Approximation

Consider function $f(x) : [0, \pi] \to [0, 1]$

$$f(x) = \sin(x)$$

Piecewise Linear Functions



Historical Comments on ReLU

- $\max(0, x)$ has a long history in statistics
- It is called linear spline basis (or hinge function)
- Bias b: in $\max(0, x + b)$ is called knot
- One hidden layer NN with ReLU is a free knot linear spline
    - Studied for $50+$ years
- In general, one considers polynomial spline bases

$$\max(0, x^k), \quad k \geq 0$$

e.g., see Chapter 5 of The Elements of Statistical Learning book by Hastie et al.

Constructive Extension to Multivariate Functions

- How about multivariate functions $f(x) : [0,1]^d \to \mathbb{R}$?
- Can we find a constructive uniform approximation?
  - Use simple functions in many dimensions.
  - Fourier series to the rescue (idea from Chen,Chen&Liu 1992 [CCL92])

Consider periodic functions $f : \mathbb{R}^d \to \mathbb{R}$ with unit period in all directions, i.e.,

$$f(\boldsymbol{x} + \boldsymbol{m}) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \mathbb{R}^d, \boldsymbol{m} \in \mathbb{Z}^d$$

Equivalently, these functions can be considered on d-torus:
$T^d = S \times \cdots S = (S)^d$, where S is a circle of circumference one.

- Note the functions on $[0,1]^d$ can be extended to periodic functions on T^d.
- Let $C(T^d)$ and $C([0,1]^d)$ be continuous real-valued functions on unit d-torus and d-cube, respectively.
- Note that $C(T^d) \subset C([0,1]^d)$, but $C([0,1]^d) \not\subset C(T^d)$ the inference $C([0,1]^d) \subset C(T^d)$ was incorrectly used in [CCL92]

Finite Series Multidimensional Fourier Approximation

- Bochner-Riesz means, $R, \alpha \geq 0$,

$$\hat{f}_R(\boldsymbol{x}) := \sum_{\boldsymbol{m}:\|\boldsymbol{m}\|_2<R, \boldsymbol{m}\in\mathbb{Z}^d} \left(1 - \frac{\|\boldsymbol{m}\|_2^2}{R^2}\right)^\alpha a_{\boldsymbol{m}} e^{2\pi i \boldsymbol{m}\cdot\boldsymbol{x}} \tag{4}$$

Then, the following uniform convergence theorm can be found in Chapter 7 of Stein&Weiss: see Corollary 2.15 on p. 256 and Theorem 2.11 (b) on p. 253.

Theorem If $f \in C(T^d)$ and $\alpha > (d-1)/2$, then $\hat{f}_R \to f$ uniformly, as $R \to \infty$, i.e., for any $\epsilon > 0$, there is $R > 0$, such that

$$\sup_{\boldsymbol{x}\in T^d} |f(x) - \hat{f}_R(x)| < \epsilon.$$

- Note that when f is real valued, we can ignore the imaginary part in Equation (4), and write ($a_{\boldsymbol{m}} = a_{\boldsymbol{m}R} + i a_{\boldsymbol{m}I}$)

$$\hat{f}_R(\boldsymbol{x}) := \sum_{\boldsymbol{m}:\|\boldsymbol{m}\|_2<R} \left(1 - \frac{\|\boldsymbol{m}\|_2^2}{R^2}\right)^\alpha (a_{\boldsymbol{m}R}\cos(2\pi i \boldsymbol{m}\cdot\boldsymbol{x}) - a_{\boldsymbol{m}I}\sin(2\pi i \boldsymbol{m}\cdot\boldsymbol{x})) \tag{5}$$

Fourier Series Recipe for NN Approximation

Consider continuous multivariable functions ($C(T^d)$). We can approximate these function using the following 2 step procedure:

1. Approximate uniformly with NNS 1D functions $\cos(u)$ and $\sin(u)$ for $|u| \leq 2\pi\sqrt{d}R$. (Note that $\boldsymbol{x} \in [0,1]^d$ and $\|\boldsymbol{x}\|_1 \leq \sqrt{d}\|\boldsymbol{x}\|_2$ imply $|\boldsymbol{m} \cdot \boldsymbol{x}| \leq \sum |m_i| = \|\boldsymbol{m}\|_1 \leq \sqrt{d}\|\boldsymbol{m}\|_2 \leq \sqrt{d}R$.)
2. Replace the NN approximation of $\sin / \cos$ into the Fourier approximation Equation (5)

Complexity: The preceding 2-step approximation has exponential complexity in d since the number of summands in Equation (5) is $\sum_{\|\boldsymbol{m}\|_2 < R} 1 \approx \text{Vol}(n\text{-Ball}) = O(R^d) = O(e^{d \log R})$
Curse of dimensionality

Uniform convergence: a brute-force way to ensure uniform convergence is to make the approximation of $\sin / \cos$ in Step 1 above very precise with an error smaller than ϵ / R^d

More General Existence Results

- Consider multivariate functions $f(x) : [0,1]^d \to \mathbb{R}$ (or some other compact domain)
- We will look at the results of Cybenko (1989) and Hornik (1991)
- These results are more general, but approximations are not explicitly constructed.
  - Instead, proofs by contradiction: elegant, but less intuitive
- Involve non-elementary mathematics: functional analysis

Results by Cybenko

Theorem (Cybenko (1989)) Let σ be a continuous sigmoidal function (recall $\lim_{x \to -\infty} \sigma(x) = 0$ and $\lim_{x \to \infty} \sigma(x) = 1$ and pick an $f \in C([0,1]^d)$. Then, the set of approximation functions of the form

$$\hat{f}(x) := \sum_j \alpha_j \sigma(< \boldsymbol{w}_j, \boldsymbol{x} > + b_j)$$

is dense in $C([0,1]^d)$ with the metric $d(f,g) = \sup |f(x) - g(x)|$, $f, g \in C([0,1]^d)$, i.e., for any $f \in C([0,1]^d)$ and $\epsilon > 0$, there exists a NN approximation function $\hat{f}(x)$, such that

$$\sup_{x \in [0,1]^d} |f(x) - \hat{f}(x)| < \epsilon.$$

Few Results From Functional Analysis

Hahn-Banach Extension Theorem If X is a normed vector space with linear subspace M and $x_0 \in X \backslash \overline{M}$, then there exists a continuous linear map $L : X \to \mathbb{R}$ with $L(x) = 0$ for all $x \in M, L(x_0) = 1$, and $\|L\| \leq d(M, x_0)$.
(E.g., see Theorem 3.5, p. 60 in Functional Analysis, 2nd ed., by Rudin.)

Use of this theorem in our context:

- Proof by contradiction: consider the subspace M given by $\{\hat{f}(\boldsymbol{x}) = \sum \alpha_j \sigma(< \boldsymbol{w}_j, \boldsymbol{x} > + b_j)\}$, and assume that its closure $\overline{M}$ is not the entire space of functions $C([0,1]^d)$.
- We conclude that there exists a continuous linear map L on our function space that restricts to 0 on $\overline{M}$ but is not identically zero.
- Hence, to prove the desired result, it suffices to show that any continuous linear map L that is zero on M must be the zero map, implying contradiction.

Few Results From Functional Analysis

Riesz Representation Theorem A bounded linear functional $L : C(X) \to \mathbb{R}$ can be expressed as

$$L(f) = \int f \, d\mu(x),$$

where μ is a finite signed measure supported on X and $f \in C(X)$.

Simple motivating example in 1D: Consider functions of one variable, $C([0,1])$, and $\sigma(x) = 1_{\{x>0\}}$.
Then, if the space spanned by $\hat{f}$, call it M, does not approximate all functions in $C([0,1])$, then there exists a linear operator L on $C([0,1])$, which is not identically zero.
But, by Riesz Theorem, for any $0 \le a < b \le 1$, and $f = 1_{\{x-a>0\}} - 1_{\{x-b>0\}} \in M$

$$L(f) = \int (1_{\{x-a>0\}} - 1_{\{x-b>0\}}) \, d\mu(x) = \int_a^b d\mu(x) = 0$$

Hence, $\mu \equiv 0$, which is a contradiction.

- The difficulty in the general proof is to extend this argument to many dimensions and general sigmoidal functions, σ.

Cybenko's Proof: Outline

Proof by contradiction:

- Consider the subspace M given by $\{\sum \alpha_j \sigma(< \boldsymbol{w}_j, \boldsymbol{x} > + b_j)\}$
- Assume that its closure $\overline{M}$ is not the entire space of functions $C([0,1]^d$. (If $\overline{M} = C([0,1]^d$, then we are done.)
- Hence, by Hahn-Banach Theorem, there exists a continuous linear map L on our function space $C([0,1]^d$ that restricts to 0 on $\overline{M}$ but is not identically zero.
- Then, by Riesz Representation Theorem, this linear functional can be expressed as integral, for any $f \in C([0,1]^d)$

$$L(f) = \int f \, d\mu(x)$$

- Key difficulty is to prove that, since $\sigma(< \boldsymbol{w}, \boldsymbol{x} > + b) \in M$,

$$\int_{[0,1]^d} \sigma(< \boldsymbol{w}, \boldsymbol{x} > + b) d\mu(\boldsymbol{x}) = 0 \quad \Rightarrow \quad \mu \equiv 0,$$

which implies $L \equiv 0$ on entire $C([0,1]^d)$, resulting in contradiction.

Lemma 1 in Cybenko

Lemma 1 Let μ be finite, signed measure on $[0,1]^d$. For any bounded, sigmoidal function, σ, if, for all $\boldsymbol{w} \in \mathbb{R}^d, b \in \mathbb{R}$

$$\int_{[0,1]^n} \sigma(<\boldsymbol{w},\boldsymbol{x}> + b) d\mu(x) = 0 \quad \Rightarrow \quad \mu \equiv 0$$

Proof Recall the step function approximation: as $\lambda \to \infty$, and then $\phi \to \infty$

$$\begin{aligned}\sigma(\lambda(<\boldsymbol{w},\boldsymbol{x}> + b) + \phi) &\to 1_{\{<\boldsymbol{w},\boldsymbol{x}>+b>0\}} + \sigma(\phi) 1_{\{<\boldsymbol{w},\boldsymbol{x}>+b=0\}} \\ &\to 1_{\{<\boldsymbol{w},\boldsymbol{x}>+b\geq 0\}}\end{aligned}$$

Hence, by Dominated Convergence Theorem, as $\lambda \to \infty$, and then $\phi \to \infty$

$$\begin{aligned}0 &= \int_{[0,1]^d} \sigma(\lambda(<\boldsymbol{w},\boldsymbol{x}> + b)) d\mu(x) = \int_{[0,1]^n} 1_{\{<\boldsymbol{w},\boldsymbol{x}>+b>0\}} d\mu(x) \\ &= \mu(\{\boldsymbol{x} :<\boldsymbol{w},\boldsymbol{x}> + b \geq 0\}),\end{aligned}$$

i.e., μ is zero on all half-spaces. (If μ were positive, we would be done.)

Lemma 1 in Cybenko

Proof - continued Since indicator (simple) functions are dense in $L^\infty(\mathbb{R})$ (bounded functions: $\mathbb{R} \to \mathbb{R}$), we conclude that the linear operator

$$F(h) = \int_{[0,1]^d} h(< \boldsymbol{w}, \boldsymbol{x} >) d\mu(\boldsymbol{x}) \equiv 0$$

for any $h \in L^\infty(\mathbb{R})$
Hence, by choosing h to be $\sin(< \boldsymbol{w}, \boldsymbol{x} >)$ and $\cos(< \boldsymbol{w}, \boldsymbol{x} >)$, we get

$$\int_{[0,1]^d} (\cos(< \boldsymbol{w}, \boldsymbol{x} >) + i \sin(< \boldsymbol{w}, \boldsymbol{x} >)) d\mu(\boldsymbol{x})$$
$$= \int_{[0,1]^d} \exp(i < \boldsymbol{w}, \boldsymbol{x} >) d\mu(\boldsymbol{x}) = 0$$

for all $\boldsymbol{w}$.
Finally, Fourier transform of μ being zero, implies

$$\mu \equiv 0.$$

This completes the proof of the lemma, and, in view of the prior outline, the proof of Theorem 2 in Cybenko.

Hornik's Extensions

Hornik follows the general plan in Cybenko, which we outlined earlier. The key technical result is the generalization of the preceding Lemma 1 of Cybenko to allowing the activation functions, $\sigma(x)$, to be bounded and nonconstant, not necessarily sigmoidal.

Theorem 5 (Hornik) Let μ be finite, signed measure on $[0,1]^d$. For any bounded and nonconstant σ, if, for all $\boldsymbol{w} \in \mathbb{R}^d, b \in \mathbb{R}$

$$\int_{[0,1]^n} \sigma(<\boldsymbol{w}, \boldsymbol{x}> + b) d\mu(x) = 0 \quad \Rightarrow \quad \mu \equiv 0$$

Proof (sketch): As in Cybenko, the key idea is to show that the Fourier transform of μ is identically zero, which implies $\mu \equiv 0$.

We want to show that, for any $\boldsymbol{w}, b$

$$\int_{\mathbb{R}^n} \sigma(<\boldsymbol{w}, \boldsymbol{x}> + b) d\mu(\boldsymbol{x}) = 0 \quad \Rightarrow \quad \mu \equiv 0.$$

First, we reduce the integration from $\mathbb{R}^d$ to $\mathbb{R}$, by defining measures $\mu_{\boldsymbol{w}}$ on $\mathbb{R}, B \subset \mathbb{R}$, as

$$\mu_{\boldsymbol{w}}(B) = \mu(\boldsymbol{x} :< \boldsymbol{w}, \boldsymbol{x} > \in B)$$

Hornik's Extensions

and observe that the prior integral becomes

$$\int_{\mathbb{R}^d} \sigma(\lambda < \boldsymbol{w}, \boldsymbol{x} > +b) d\mu(\boldsymbol{x}) = \int_{\mathbb{R}} \sigma(\lambda t + b) d\mu_{\boldsymbol{w}}(t) = 0$$

Moreover, if we can show that $\mu_{\boldsymbol{w}} \equiv 0$ for each $\boldsymbol{w}$, then $\mu \equiv 0$ ("a measure is defined by all of its projections"), since then

$$\hat{\mu}(\boldsymbol{w}) = \int_{\mathbb{R}^d} \exp(i < \boldsymbol{w}, \boldsymbol{x} >) d\mu(\boldsymbol{x}) = \int_{\mathbb{R}} \exp(it) d\mu_w(t) = 0 \Rightarrow \hat{\mu} = 0 \Rightarrow \mu = 0.$$

(Note that we used the finiteness of μ here.)
Hence, the goal is to find arguments that justify the preceding equation.

Hornik's Extensions

To this end, use the convolution trick (that also uses the finiteness of μ).
By convolving $\mu_{\boldsymbol{w}}(t)$ with a Gaussian $v(t) = e^{-t^2}$, we obtain a measure that has a density, letting us work with Lebesgue measure. $\mu_{\boldsymbol{w}} * v$ is also finite.
Next,

$$\begin{aligned} 0 &= \int_{\mathbb{R}} \left[\int_{\mathbb{R}} \sigma(\lambda t + (b + \lambda s)) d\mu_{\boldsymbol{w}}(t) \right] v(s) ds \\ &= \int_{\mathbb{R}} \int_{\mathbb{R}} \sigma(\lambda(t+s) + b) d\mu_{\boldsymbol{w}}(t) v(s) ds d\mu_{\boldsymbol{w}}(t) \\ &= \int_{\mathbb{R}} \sigma(\lambda t + b) d(v * \mu_{\boldsymbol{w}})(t); \end{aligned}$$

let $h(t) = v * \mu_{\boldsymbol{w}})(t)$ is the convolution of $\mu_{\boldsymbol{w}}(t)$ and $v(t)$.
Then, using the results from abstract Fourier analysis, one shows that $h \equiv 0$. (see: Fourier Analysis on Groups, Rudin 1967)
This implies that the Fourier transform $\hat{h} = \hat{v}\hat{\mu}_{\boldsymbol{w}} = 0$.
Since, $\hat{v}$ has no zeros, this results in $\hat{\mu}_{\boldsymbol{w}} \equiv 0 \Rightarrow \hat{\mu} \equiv 0 \Rightarrow \mu \equiv 0$, concluding the proof of Theorem 5 in Hornik.

Hornik's Extensions

Using Theorem 5 with more general activation functions, Hornik obtains the following results:

- **Theorem 1** If σ is unbounded and nonconstant, then NN approximations are dense in $L^p(\mu)$ for all finite measures μ on $\mathbb{R}^d$. $L^p(\mu), p \geq 1$, is the space of functions f with $\int |f|^p d\mu < \infty$ and distance metric $d(f,g) = \left(\int |f-g|^p d\mu\right)^{1/p}$.
- **Theorem 2** If σ is continuous, bounded and nonconstant, then NN approximation are dense in the space of all continuous functions, $C(X)$, with compact domain $X \subset \mathbb{R}^k$ and supremum distance $d(f,g) = \sup_{x \in X} |f(x) - g(x)|$. (This is a full generalization of Theorem 2 in Cybenko.)
- **Theorems 3&4** Extend results to Sobolev spaces under the $\ell_p, 1 \leq p < \infty$ and supremum norm.

  Sobolev spaces contain function that have up to m derivatives and distance is measured between functions and their derivatives.

Reading on the Universal Approximation Results

Shallow (1 hidden layer) networks are universal approximators

- Constructive proofs:
  - A Constructive Proof and An Extension of Cybenko's Approximation Theorem, by Chen, Chen & Ruey-wen Liu, 1992. (Note: Not more general than Cybenko: requires continuity of periodic function extensions.)
  - Constructive Approximation by Superposition of Sigmoidal Functions, by Costarelli and Spigler, 2013. (1D&2D)
- General proofs:
  - Approximations by superpositions of sigmoidal functions, by Cybenko, 1989.
  - Approximation capabilities of multilayer feedforward networks, by Hornik, 1991.

Reading For Next Class

Bounds on NN approximations:

- Universal approximation bounds for superpositions of a sigmoidal function, by Barron 1993.
- Approximation theory of the mlp model in neural networks, by Pinkus, 1999.

Expressive Power Depth Depth separation results: Can deep neural networks of depth $(d+1)$ express functions much more efficiently in terms of the number of neurons compared to networks of depth d?

- Representation Benefits of Deep Feedforward Networks, by Telgarsky, 2015.
- Error bounds for approximations with deep ReLU networks, by Yarotsky, 2017.
- WHY DEEP NEURAL NETWORKS FOR FUNCTION APPROXIMATION?, Lianf and Srikant, 2017.
- NEW ERROR BOUNDS FOR DEEP RELU NETWORKS USING SPARSE GRIDS, by Montanelli and Du, 2018.
- The Power of Depth for Feedforward Neural Networks, by Eldan and Shamir, 2016.

General references on deep learning:

[UML] Chapter 14: Stochastic Gradient Descent
Chapter 20: Neural Networks

[DL] Chapter 6: Deep Feedforward Networks

Software: Download R, R Studio and TensorFlow
Or, equivalent for Python

If you would like to refresh or learn the concepts form real analysis, you could check Chapter 2 on basic topology concepts and Chapter 4 for continuity/uniform continuity in

- Principles of Mathematical Analysis, by W. Rudin.

More advanced reading on Fourier analysis

- Introduction to Fourier Analysis on Euclidian Spaces, by Stein and Weiss, 1971. (Chapter 7 is on multivariate Fourier series and Bochner-Riesz means.)
- Fourier Analysis on Groups, by W. Rudin. (Chapter 7 is used in Hornik.)