CSCE 790: Neural Networks and Their Applications

AIISC and Dept. Computer Science and Engineering
Email: vignar@sc.edu

© Vignesh Narayanan

September 19, 2023

UNIVERSITY OF
South Carolina

What function does an NN learning in a regression problem?

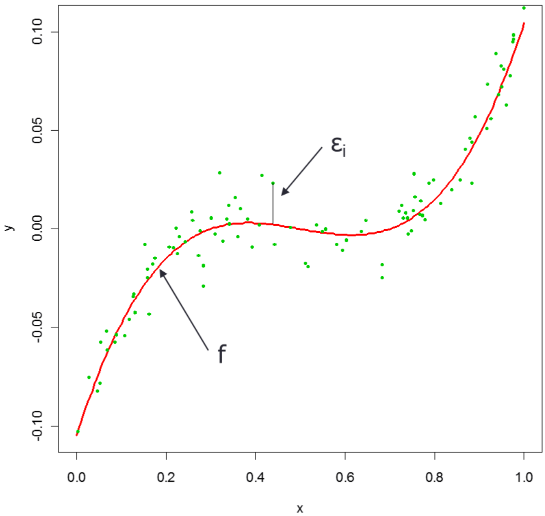


Figure: Regression

What function does an NN learning in a regression problem?

- The function f is unknown
- We estimate f for two key purposes:
  - Prediction
  - Inference
- By producing a good estimate for f where the variance of ε is not too large, we can make accurate predictions for the response variable, y, based on a new value of x
- The accuracy of a prediction for y depends on: Reducible error and Irreducible error

Regression errors

- Note that the model will not be a perfect estimate for f - the correct relationship between input-output data; this inaccuracy introduces error
- This error is reducible because we can potentially improve the accuracy of the estimated (i.e., hypothesis) model by using the most appropriate learning technique to estimate the target function f
- Even if we could perfectly estimate f, there is still variability associated with ε that affects the accuracy of predictions $=$ irreducible error

What function does an NN learning in a regression problem?

- For example, consider the average of the squared difference between the predicted and actual value of y
- $Var(\varepsilon)$ represents the variance associated with ε

$$\mathbb{E}[(y - \hat{f}(X))^2 | X = x] = \underbrace{[f(x) - \hat{f}(x)]^2}_{\text{Reducible}} + \underbrace{Var(\varepsilon)}_{\text{Irreducible}}$$

- The aim of the learning process is to minimize the reducible error
- What function does an NN learning in a classification problem?

Decision Boundaries

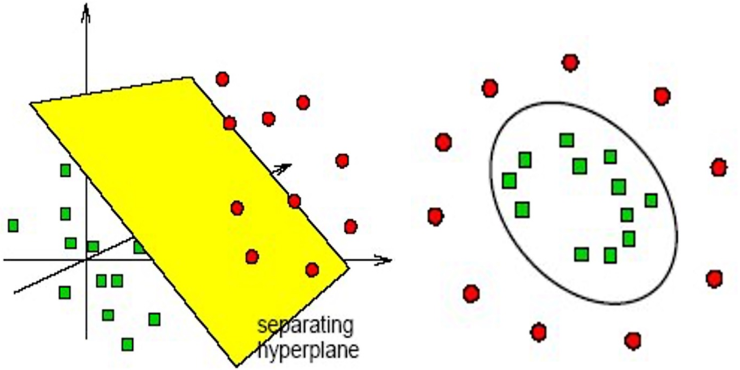


Figure: Decision boundaries - Linear and nonlinear boundaries

Revisiting simple NN - OR Logic

- We already discussed that the OR function's thresholding parameter theta is 1, for obvious reasons.
- The inputs are obviously boolean, so only 4 combinations are possible $(0, 0), (0, 1), (1, 0)$ and $(1, 1)$

Revisiting Simple NN - OR Logic

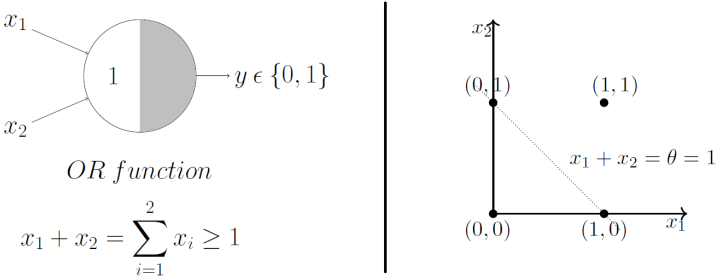
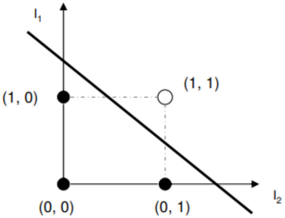
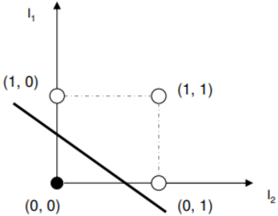


Figure: Logic-OR gate **

AND		
I_1	I_2	out
0	0	0
0	1	0
1	0	0
1	1	1



OR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	1



XOR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	0

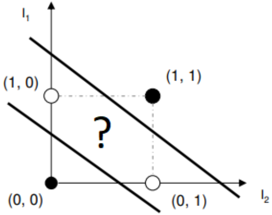


Figure: Logic gates and XOR problem

Perceptron in general?

- A 1−layer NN with two inputs x_1, x_2 and one output y is given by

$$y = \sigma(v_0 + v_1 x_1 + v_2 x_2)$$

- Let $\sigma(\cdot)$ be the symmetric hard limit
- The output space $= \{-1, 1, 0\}$
- When $y = 0$
  - $0 = v_0 + v_1 x_1 + v_2 x_2$
  - $x_2 = -\frac{v_0}{v_2} - \frac{v_1}{v_2} x_1$
- This defines a line partitioning $\mathbb{R}^2$ into two decision regions, with $y = +1$ in one region and $y = -1$ in the other region
- In the general case of n inputs x_j and m outputs y_l, the one layer NN partitions $\mathbb{R}^n$ using m hyperplanes (subspace of dimension $n - 1$)

Separability

- Linear separability (for boolean functions): There exists a line (plane) such that all inputs which produce a 1 lie on one side of the line (plane) and all inputs which produce a 0 lie on other side of the line (plane).

What NN learns?

Visualization

Underfitting vs Overfitting

- There are always two aspects to consider when designing a learning algorithm:
  - Try to fit the data well
  - Be as robust as possible
- The predictor that you have generated using your training data must also work well on new data.
- When we create predictors, usually the simpler the predictor is, the more robust it tends to be in the sense of being able to be estimated reliably.
- On the other hand, the simple models do not fit the training data aggressively.

Underfitting vs Overfitting

- If you try to fit the data too aggressively, then you may over-fit the training data.
- This means that the predictors works very well on the training data, but is substantially worse on the unseen test data.

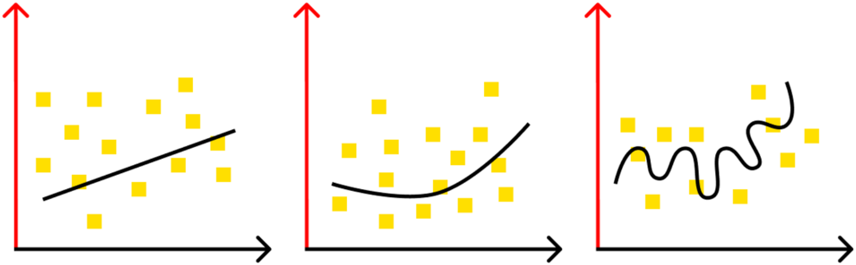


Figure: Underfitting vs Overfitting

Bias vs Variance

- Bias $\rightarrow$ how good the predictor is, on average; tends to be smaller with more complicated models
- Variance $\rightarrow$ tends to be higher for more complex models
- Simple model (e.g., just linear term) introduces (model) bias
- Highly complex model introduces high variance

Training vs Testing Error

- Training error $\rightarrow$ reflects whether the data fits well
- Testing error $\rightarrow$ reflects whether the predictor actually works on new data

Generalization?

- The out-of-sample error E_{out} measures how well our training on D has generalized to data that we have not seen before.
- E_{out} is based on the performance over the entire input space X.
- Intuitively, if we want to estimate the value of E_{out} using a sample of data points, these points must be 'fresh' test points that have not been used for training.
- The in sample error E_{in}, by contrast, is based on data points that have been used for training.

Generalization

- The value of E_{in} does not always generalize to a similar value of E_{out}.
- Generalization is a key issue in learning.
- One can define the generalization error as the discrepancy between E_{in} and E_{out} .
- Universal approximation theorem warrants $E_{in} \to 0$ as number of neurons in the hidden layers $\to \infty$
- For more, check: Learning from Data: A Short Course, by Hsuan-Tien Lin, Malik Magdon-Ismail, and Yaser Abu-Mostafa

Interpolation

Definition

Interpolation occurs for a sample x whenever this sample belongs to the convex hull of a set of samples $X \triangleq \{x_1, \ldots, x_N\}$, if not, extrapolation occurs.

Theorem

Given a $d-$dimensional dataset $X \triangleq \{x_1, \ldots, x_N\}$ with i.i.d. samples uniformly drawn from an hyperball, the probability that a new sample x is in interpolation regime has the following asymptotic behavior

$$\lim_{d \to \infty} \underbrace{p(x \in Hull(X))}_{Interpolation} = \begin{cases} 1 \iff N > d^{-1} 2^{\frac{d}{2}} \\ 0 \iff N < d^{-1} 2^{\frac{d}{2}} \end{cases} \quad (1)$$

Learning in High Dimension Always Amounts to Extrapolation

Convex Set

Definition

A subset $C \subset \mathbb{R}^n$ is a convex set if $\alpha x + (1-\alpha)y \in C$, $\forall x, y \in C$, $\forall \alpha \in [0,1]$.

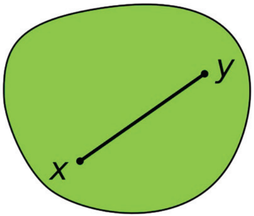


Figure: Convex Set

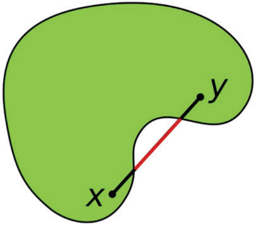


Figure: Non-Convex Set

Figure: (a) Illustration of a convex set which looks somewhat like a deformed circle.

Figure: (b) Illustration of a non-convex set which looks somewhat like a boomerang.

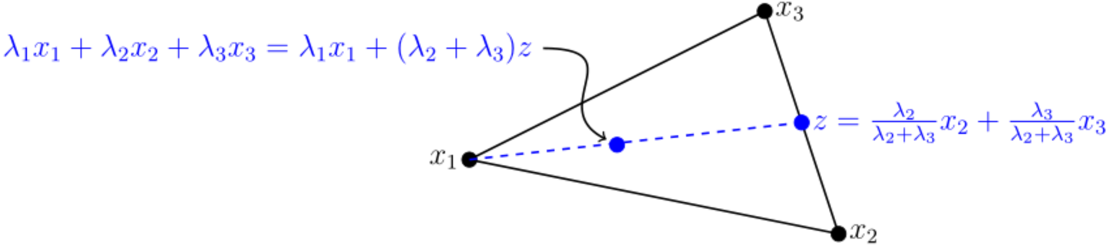
Convex sets - Example

Example (Convex sets)

1. A hyperplane $H = \{x \in \mathbb{R}^n : p^T x = c\}$ for some $p \in \mathbb{R}^n$, $p \neq 0$, and $c \in \mathbb{R}$, for example, a plane in $\mathbb{R}^3$, $p_1 x + p_2 y + p_3 z = 1$.
2. Half space $H^+ = \{x \in \mathbb{R}^n : p^T x \geq c\}$ or $H^- = \{x \in \mathbb{R}^n : p^T x \leq c\}$.

Simplex - Building blocks of hulls

The set of all convex combinations $\sum_{i=1}^{3} \lambda_i x_i$ of $x_1, x_2, x_3 \in \mathbb{R}^n$ with $\lambda_1 + \lambda_2 + \lambda_3 = 1$ is the triangular region determined by x_1, x_2, x_3 (formed between vertices x_1, x_2, x_3).



Convex Hull

More generally, the set of all convex combinations $\sum_{i=1}^{k} \lambda_i x_i$ of k vectors $x_1, \ldots, x_k \in \mathbb{R}^n$ is the convex polyhedral region determined by $x_1, \ldots, x_k$ (the so-called convex Hull, the intersection of all convex sets containing x_i, $i = 1, \ldots, k$).

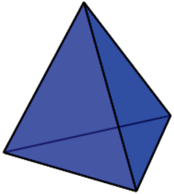


Figure: Illustration of a tetrahedron that is a convex combination of four vectors.

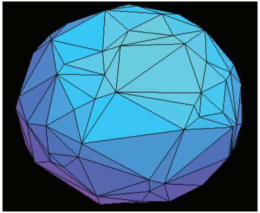
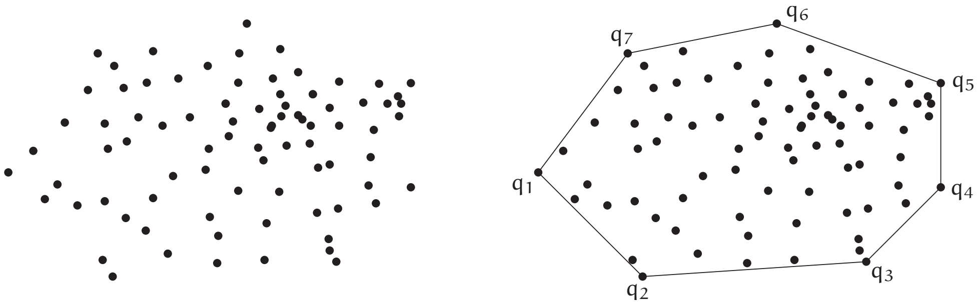


Figure: A convex hull of 100 random uniform points on a sphere.

Convex Hull



(a) Input. (b) Output.

Figure: Convex Hull of a set of points in $\mathbb{R}^2$.

Convex Hull

Given an arbitrary set S in $\mathbb{R}^n$, different convex sets can be generated from S. In particular, we discuss below the convex hull of S.

Definition (Convex hull)

Let S be an arbitrary set in $\mathbb{R}^n$. The convex hull of S, denoted $\text{conv}(S)$, is the collection of all convex combinations of S. In other words, $x \in \text{conv}(S)$ if and only if x can be represented as

$$x = \sum_{j=1}^{k} \lambda_j x_j, \quad \sum_{j=1}^{k} \lambda_j = 1,$$
$$\lambda_j \geq 0 \quad \text{for} \quad j = 1, \ldots, k,$$

where k is a positive integer and $x_1, \ldots, x_k \in S$.

Convex optimization (for completeness)

- Convex cost function
- Constraints represented by convex functions
- Convex constraints $\implies$ constraint set is convex
- $\rightarrow$ Convex optimization problem