Deep Learning course

E. **Francisco** Roman-Rangel
edgar.roman@alumni.epfl.ch

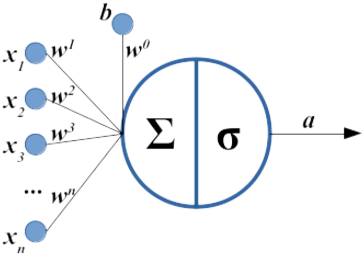
Session 5 – Multi-layer perceptron (MLP)

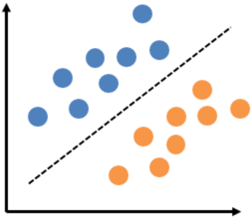
CInC-UAEM. Cuernavaca, Mexico. September 15th, 2018.

Outline

Perceptron

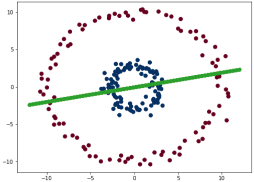
- The simplest ANN, inspired by the neuron.
- Often using step or sigmoid activation functions $\sigma(\cdot)$.
- Linear classifier. Good enough for binary classification.
- Also good for linear regression.

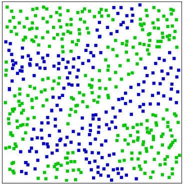


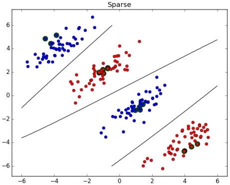


Perceptron

- Limited when dealing with non-linear separations.
- Also for multi-class problems (one perceptron per class).
- Difficult to guess whether a problem is linearly separable.
  e.g., Iris dataset classification performance.







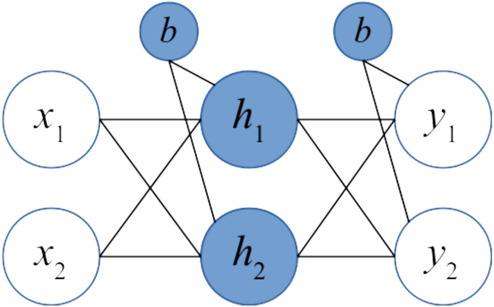
Perceptron

- Limited when dealing with non-linear separations.
- Also for multi-class problems (one perceptron per class).
- Difficult to guess whether a problem is linearly separable.
  e.g., Iris dataset classification performance.

Outline

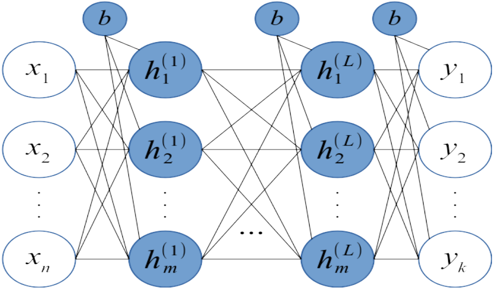
MLP Network

- Arrangement of artificial neurons (a.k.a., units, nodes).
- Arranged within consecutive layers.
- Vanilla: input layer, hidden layers, output layer.
- Layers provide a hierarchically representation.



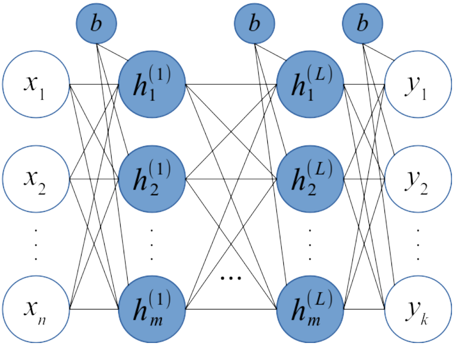
MLP Network

- Architecture: specific arrangement of the layers and nodes.
- Size: number of nodes in the model.
- Width: number of nodes in a specific layer.
- Depth: number of layers in a neural network.
- Capacity: function that the network can learn.



Number of parameters

Fully connected network.



$$n\_paramL = \Omega^{(L)}.shape[0] \times \Omega^{(L)}.shape[1] + \Omega^{(L)}.shape[0]$$

Bias terms

- One per unit.
- One node ($b^{(l)} = 1$) per layer, and
- multiple weights: $\omega^{(l)}_{[m \times 1]}$.

How many layers and nodes?

Who knows?

- Input layer: one placeholder per feature.
- First hidden layer: one node per hyperplane.
- Output layer: one node per class.
- Intermediate layers: as many nodes as needed to (combine hyperplanes, induce sparsity, be robust, avoid overfitting, have enough capacity, reduce dimensionality, etc).
- Same for the number of intermediate layers: special attention on capacity.

How many layers and nodes?

- Experimentation.
- Intuition.
- Keep it deep.
- Imitate the masters.

Outline

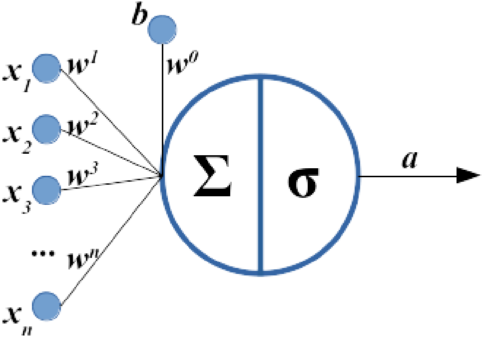
Perceptron

Multi-layer perceptron

Activations

Activation function

Remember activation functions inside each unit.

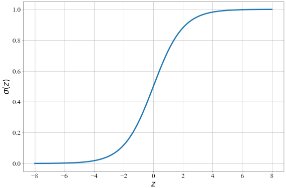


Activation function

- Non linearity.
- Continuously differentiable.
- Finite range.
- Monotonic.
- Approximate identity near the origin.

Logistic sigmoid (sigmoid)

$$\sigma(z) = \frac{1}{(1 + \exp^{-z})}$$



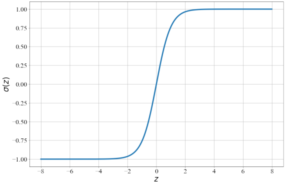
- Positive real numbers within $[0.0, 1.0]$.
- Large negative numbers $\rightarrow 0$, and large positive numbers $\rightarrow 1$.
- Provides a notion of probability.
- *cons*: saturates, i.e., gradient close to 0.

Derivative:

$$\sigma'(z) = \sigma(z)(1 - \sigma(z))$$

Hyperbolic tangent (tanh)

$$\sigma(z) = \tanh(z) = \frac{\exp^z - \exp^{-z}}{(\exp^z + \exp^{-z})}$$



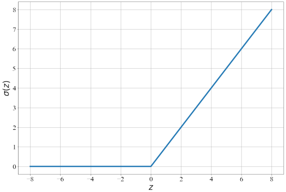
- Alternative to sigmoid.
- Real numbers within $[-1.0, +1.0]$.
- Negative numbers will remain negative.
- Zero centered.
- *cons*: saturates, i.e., gradient close to 0.

Derivative:

$$\sigma'(z) = 1 - (\tanh(z))^2$$

ReLU

$$\sigma(z) = \max(0, z)$$



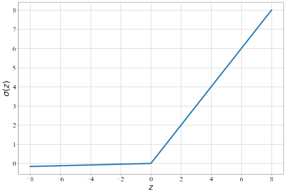
- Alternative to sigmoid.
- Positive real numbers within $[0.0, \infty)$.
- Much faster for optimization.
- *cons*: might saturate on the left.

Derivative:

$$\sigma'(z) = \begin{cases} 0, & z < 0 \\ 1, & z \leq 0. \end{cases}$$

Leaky ReLU

$$\sigma(z) = \max(0.01z, z)$$



- Minimizes the risk of saturating on the left.
- Real numbers within $[\approx 0.01z, \infty)$.
- Faster enough for optimization.

Derivative:

$$\sigma'(z) = \begin{cases} 0.01, & z < 0 \\ 1, & z \leq 0. \end{cases}$$

Soft-max

$$\sigma(z_i) = \frac{\exp(z_i)}{\sum_k \exp(z_k)}$$

- Suitable for last layer.
- Maps a vector onto a pdf.
- Real numbers within $[\approx 0.01z, \infty)$.

Derivative:

$$\sigma'(z_i) = \begin{cases} \sigma(z_i)(1-\sigma(z_i)), & i = j \\ -\sigma(z_j)\sigma(z_i), & i \neq j. \end{cases}$$

Most used activations

- ReLU [variants] (all hidden layers).
- Sigmoid (output layer for regression and binary classification).
- Softmax (output layer for multi-class classification).

Suggested readings

- http://peterroelants.github.io/posts/neural_network_implementation_intermezzo02
- DL Book. Goodfellow.

Thank you.

Q&A