Learning Deep Architectures for AI

- Yoshua Bengio

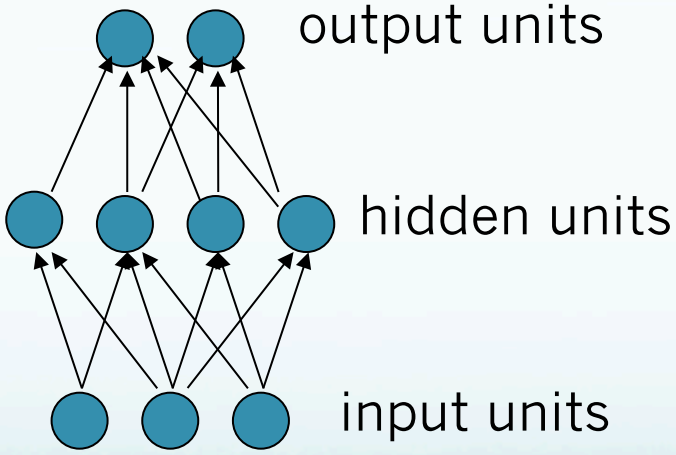
Part I

- Vijay Chakilam

Chapter 0: Preliminaries

Neural Network Models

- The basic idea behind the neural network approach is to model the response as a nonlinear function of various linear combinations of the predictors

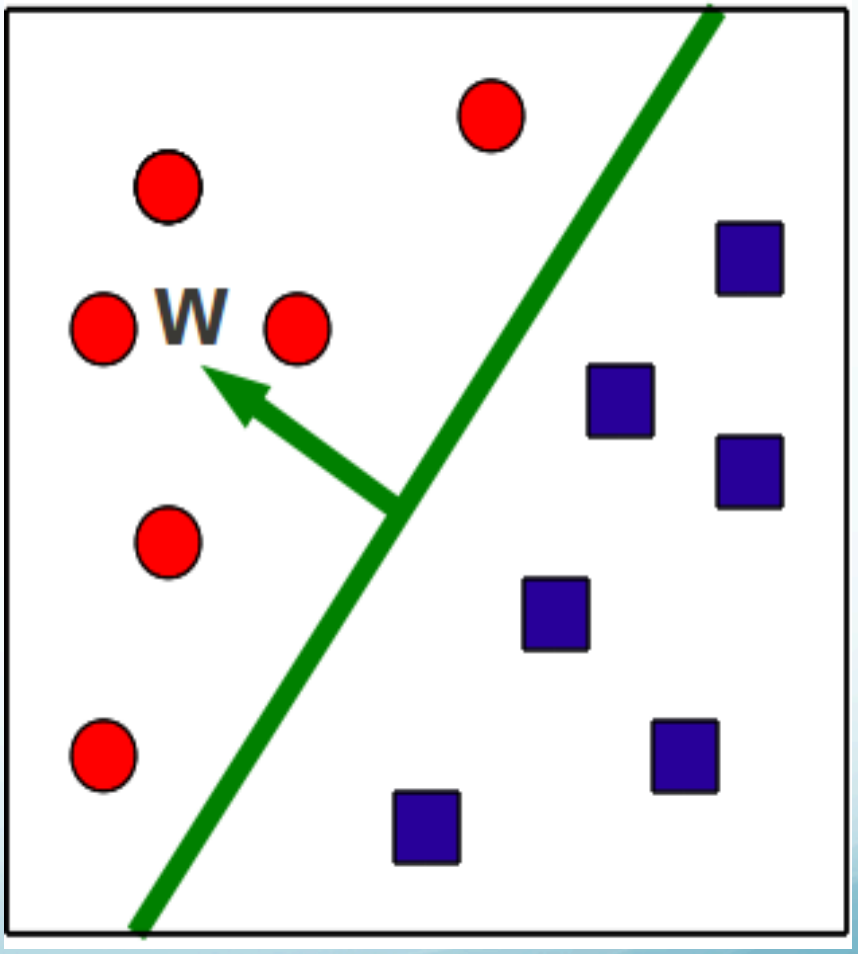


Types of Neurons

- Linear
- Binary threshold
- Rectified Linear
- Sigmoid

Perceptron

- One of the earliest algorithms for linear classification (Rosenblatt, 1958)
- Based on finding a separating hyperplane of the data



Perceptron Architecture

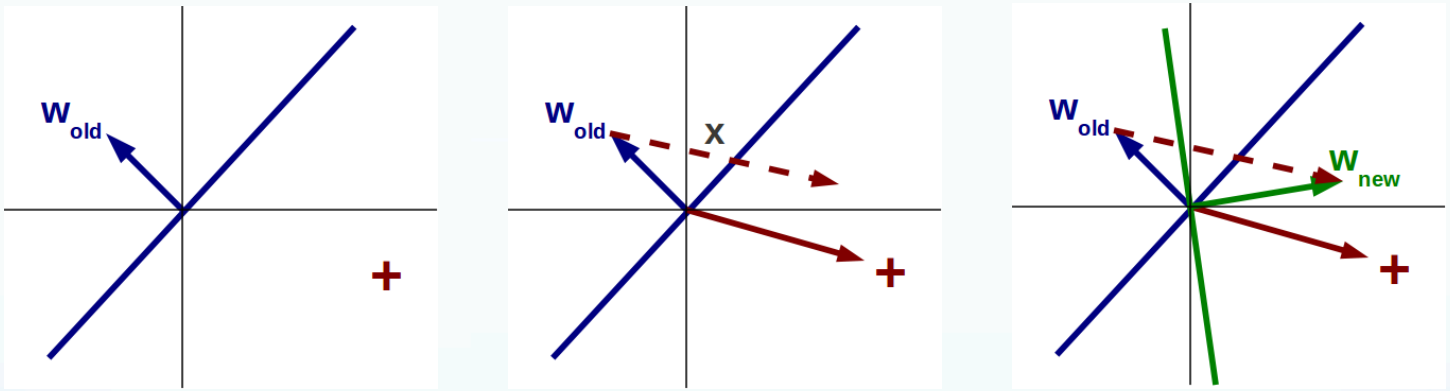
- Manually engineer features; mostly based on common sense and hand-written programs.
- Learn how to weight each of the features to get a single scalar quantity.
- If this quantity is above some threshold, decide that the input vector is a positive example of the target class.

Perceptron Algorithm

- Add an extra component with value 1 to each input vector. The "bias" weight on this component is minus the threshold. Now we can have the threshold set to zero.
- Pick training cases using any policy that ensures that every training case will keep getting picked.
  - If the output unit is correct, leave its weights alone.
  - If the output unit incorrectly outputs a negative class, add the input vector to the weight vector.
  - If the output unit incorrectly outputs a positive class, subtract the input vector from the weight vector.
- This is guaranteed to find a set of weights that gets the right answer for all the training cases if any such set exists.

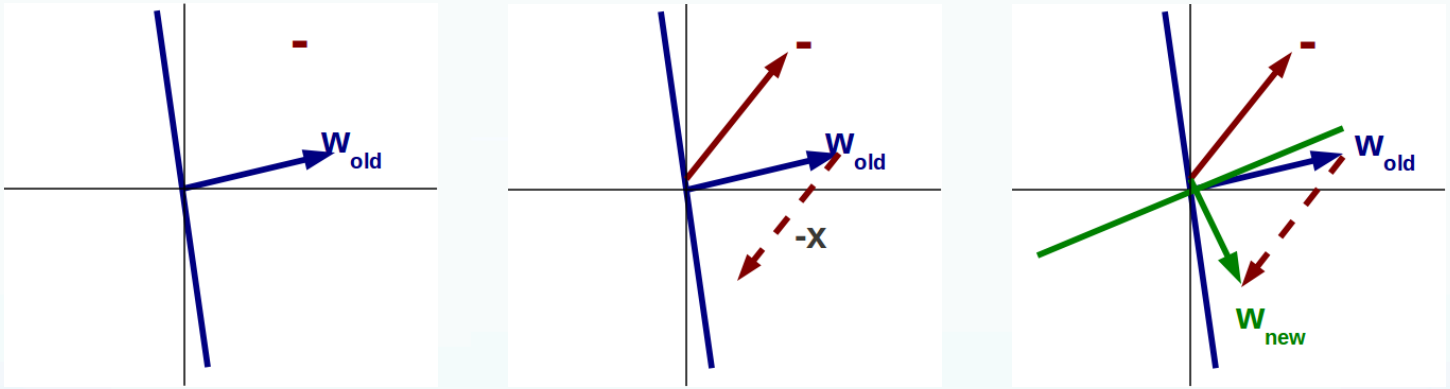
Perceptron Updates

$$w_{new} = w_{old} + x$$



Perceptron Updates

$$w_{new} = w_{old} - x$$



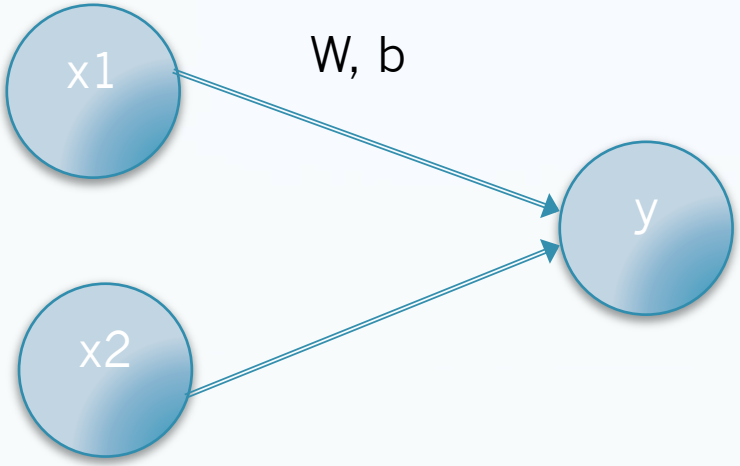
Perceptron Convergence

- Theorem (Block, 1962, and Novikoff, 1962).

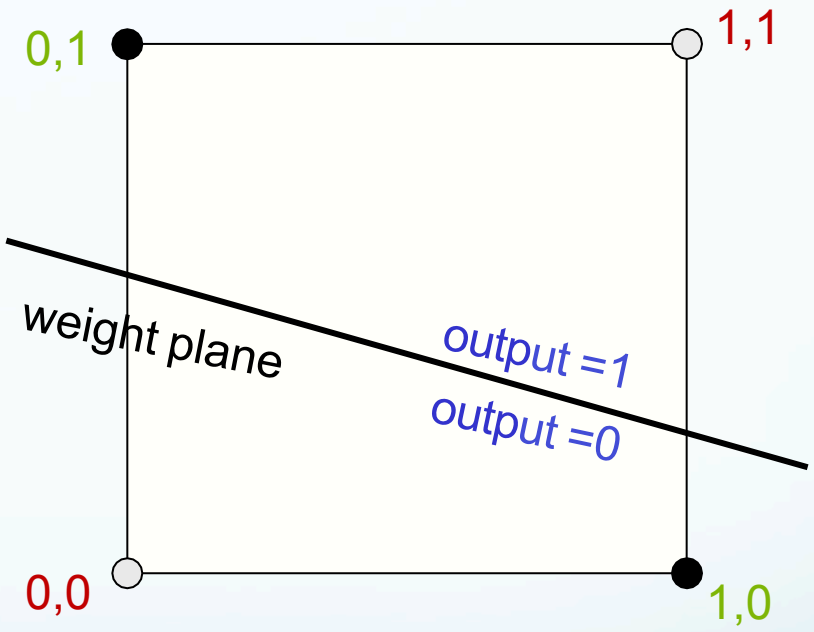
  - Let a sequence of examples $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots . (x^{(m)}, y^{(m)})$ be given. Suppose that $\| x^{(i)} \| \leq D$ for all i, and further that there exists a unit-length vector u $(\| u \|_2 = 1)$ such that $y^{(i)}.(u^T x^{(i)}) \geq \gamma$ for all examples in the sequence. Then the total number of mistakes that the algorithm makes on this sequence is at most $(D/\gamma)^2$

Chapter 1: Deep Architectures

Limitations of Perceptron

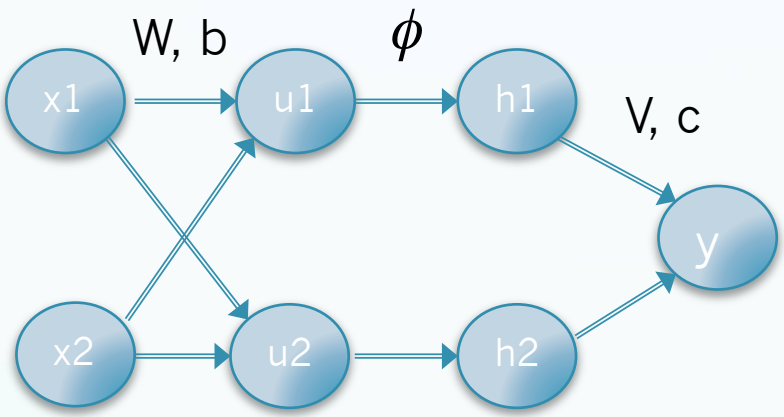


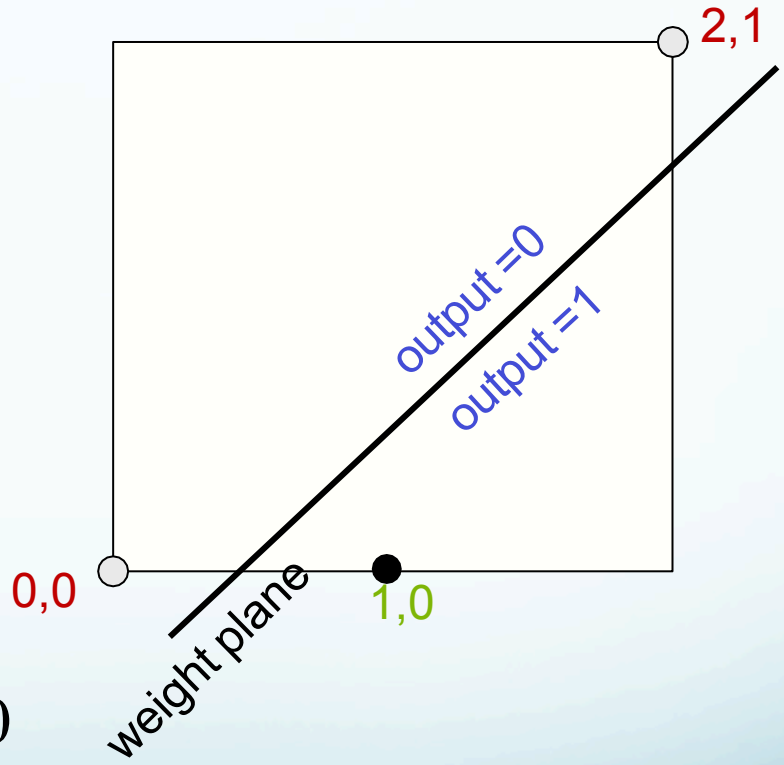
- There is no value for W and b such that the model results in right target for every example



A graphical view of the XOR problem

Learning representations





$$W = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \; b = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \; V = \begin{bmatrix} 1 \\ -2 \end{bmatrix} \text{ and } c = 0$$

Universal Approximation

- Theorem: A neural network with one hidden layer can approximate any continuous function.
- More formally, given a continuous function $f : C_n \mapsto R^m$ where C_n is a compact subset of R^n ,

$$\forall \varepsilon, \exists f_{NN}^{\varepsilon} : x \rightarrow \sum_{i=1}^{K} V_i \phi(W_i^T + b_i) + c$$

such that

$$\forall x \in C_n, \left\| f(x) - f_{NN}^{\varepsilon}(x) \right\| < \varepsilon$$

Universal Approximation

- Cybenko 1989; Hornik et al., 1989 proved the universal approximation properties for networks with squashing activation function such as logistic sigmoid.
- Leshno et al., 1993 proved the UA properties for Rectified Linear Unit activation functions.

Deep vs Shallow

- Pascanu et al. (2013) compared deep rectifier networks with their shallow counterparts.
  - For a deep model with n_0 inputs and k hidden layers of width n, the maximal number of response regions per parameter behaves as

$$\Omega\left(\left\lfloor \frac{n}{n_0} \right\rfloor^{k-1} \frac{n^{n_0-2}}{k}\right)$$

  - For a shallow model with n_0 inputs and nk hidden units, the maximal number of response regions per parameter behaves as

$$O\left(k^{n_0-1} n^{n_0-1}\right)$$

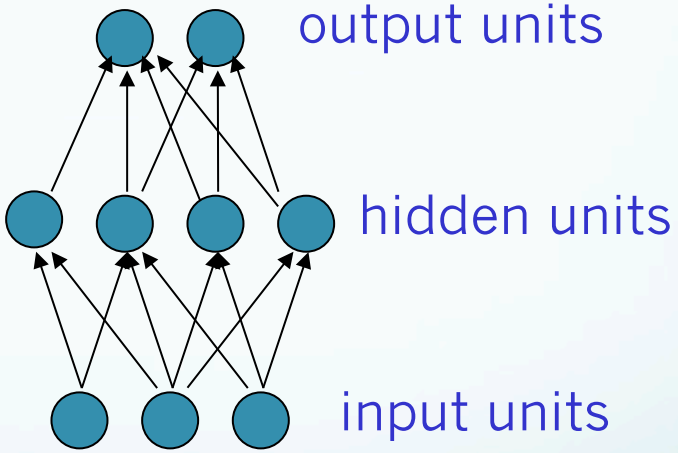
Deep vs Shallow

- Pascanu et al. (2013) showed that the deep model can generate exponentially more regions per parameter in terms of the number of hidden layers, and at least order polynomially more regions per parameter in terms of layer width n.
- Montufar et al. (2014) came up with significantly improved lower bound on the maximal linear regions.

Chapter 2: Feed-forward Neural Networks

Feed-forward neural networks

- These are the commonest type of neural network in practical applications.
  - The first layer is the input and the last layer is the output.
  - If there is more than one hidden layer, we call them "deep" neural networks.
- They compute a series of transformations that change the similarities between cases.
  - The activities of the neurons in each layer are a non-linear function of the activities in the layer below.

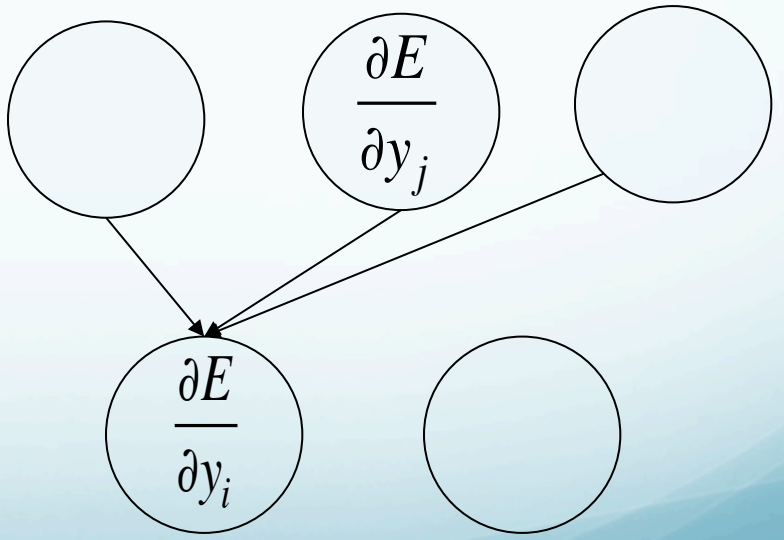


Backpropagation

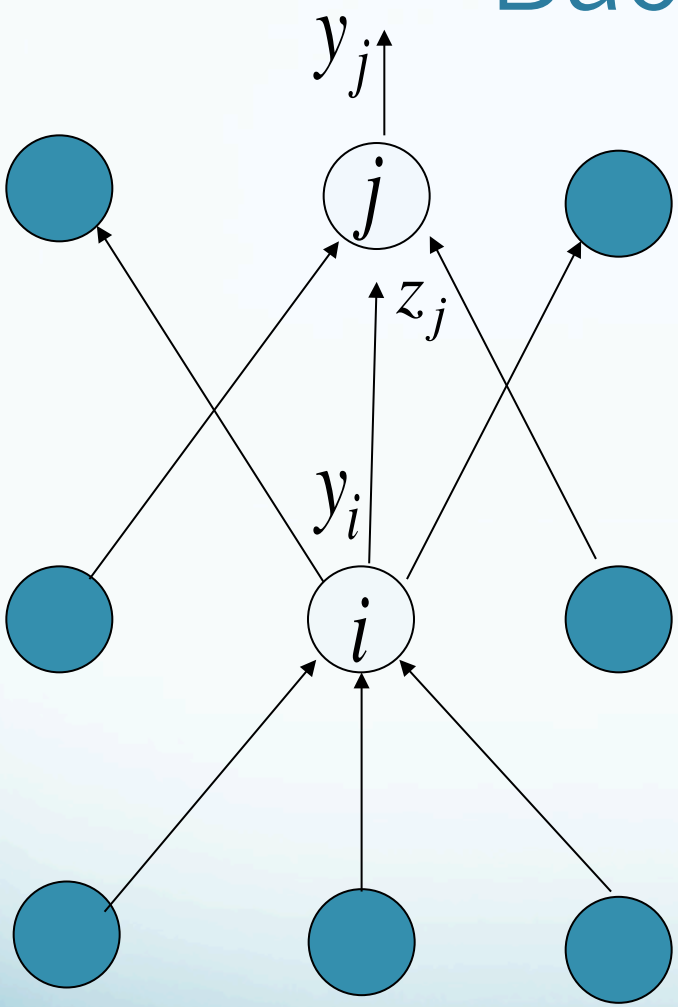
- First convert the discrepancy between each output and its target value into an error derivative.
- Then compute error derivatives in each hidden layer from error derivatives in the layer above.
- Then use error derivatives *w.r.t.* activities to get error derivatives *w.r.t.* the incoming weights.

$$E = \frac{1}{2} \sum_{j \in output} (t_j - y_j)^2$$

$$\frac{\partial E}{\partial y_j} = -(t_j - y_j)$$



Backpropagation



$$E = \frac{1}{2} \sum_{j \in output} (t_j - y_j)^2$$

$$\frac{\partial E}{\partial y_j} = -(t_j - y_j)$$

$$\frac{\partial E}{\partial z_j} = \frac{dy_j}{dz_j} \frac{\partial E}{\partial y_j} = y_j (1 - y_j) \frac{\partial E}{\partial y_j}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial z_j}{\partial w_{ij}} \frac{\partial E}{\partial z_j} = y_i \frac{\partial E}{\partial z_j}$$

$$\frac{\partial E}{\partial y_i} = \sum_j \frac{dz_j}{dy_i} \frac{\partial E}{\partial z_j} = \sum_j w_{ij} \frac{\partial E}{\partial z_j}$$

References

- Yoshua Bengio, *Learning deep architecures for AI*: http://www.iro.umontreal.ca/~bengioy/papers/ftml_book.pdf
- Geoff Hinton, *Neural networks for machine learning*: https://www.coursera.org/learn/neural-networks
- Goodfellow et al., *Deep Learning*: http://www.deeplearningbook.org/
- Montufar et al., *On the number of linear regions of Deep Networks*: https://arxiv.org/pdf/1402.1869.pdf

References

- Pascanu et al., *On the number of response regions of deep feed forward neural networks with piece-wise linear* activations: https://arxiv.org/pdf/1312.6098.pdf
- Perceptron weight update graphics: https://www.cs.utah.edu/~piyush/teaching/8-9-print.pdf
- Proof of Theorem (Block, 1962, and Novikoff, 1962). http://cs229.stanford.edu/notes/cs229-notes6.pdf