Learning Neural Networks - Part 2

Alice Gao

Lecture 9

Readings: RN 18.7, PM 7.5.

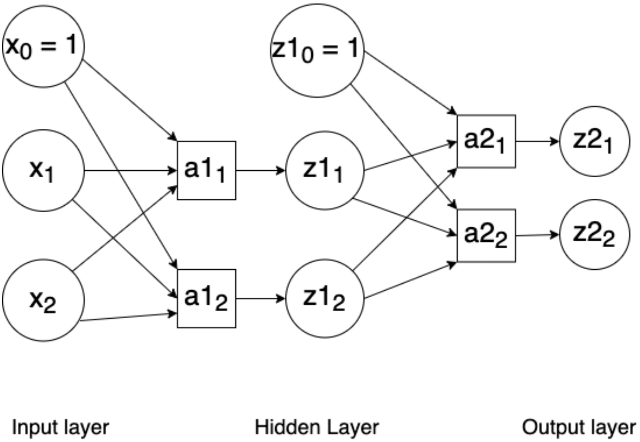
Outline

Learning Goals

By the end of the lecture, you should be able to

- Explain the steps of the gradient descent algorithm.
- Explain how we can modify gradient descent to speed up learning and ensure convergence.
- Describe the back-propagation algorithm including the forward and backward passes.
- Compute the gradient for a weight in a multi-layer feed-forward neural network.
- Describe situations in which it is appropriate to use a neural network or a decision tree.

A 2-Layer Neural Network



Gradient Descent

"Walking downhill and always taking a step in the direction that goes down the most."

- A local search algorithm to find the minimum of a function.
- Steps of the algorithm:
  - Initialize weights randomly.
  - Change each weight in proportion to the negative of the partial derivative of the error with respect to the weight.

$$W := W - \eta \frac{\partial E}{\partial W}$$

  - η is the learning rate.
  - Terminate after some number of steps
    when the error is small or when the changes get small.

Why update the weight proportional to the negative of the partial derivative?

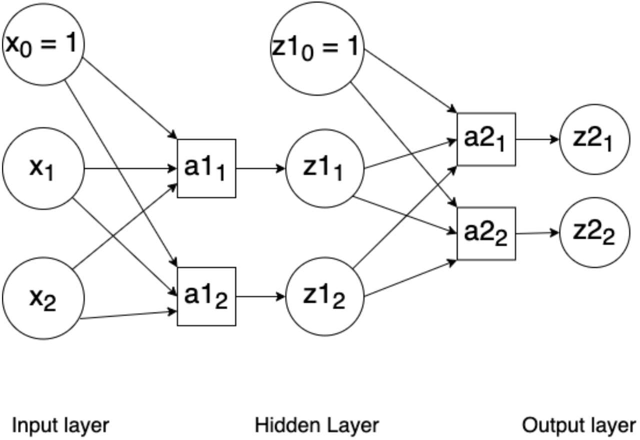
- Suppose that we want to find the minimum of $y = x^2$.
- Start with $x = x_0$.
- In what direction should we change the value of x?

- By what amount should we change the value of x?
  What is the step size?

How do we update the weights based on the data points?

- Gradient descent updates the weights after sweeping through all the examples.
- To speed up learning, update weights after each example.
  - Incremental gradient descent
  - Stochastic gradient descent
- Trade off learning speed and convergence.
  - Batched gradient descent

A 2-Layer Neural Network



The Back-propagation Algorithm

- An efficient method of calculating the gradients in a multi-layer neural network.
- There are some training examples $(\vec{x}_n, \vec{y}_n)$ and an error/loss function $E(z2, y)$. Perform 2 passes.
  - Forward pass: compute the error E given the inputs and the weights.
  - Backward pass: compute the gradients $\frac{\partial E}{\partial W2_{jk}}$ and $\frac{\partial E}{\partial W1_{ij}}$.
- Update each weight by the sum of the partial derivatives for all the training examples.

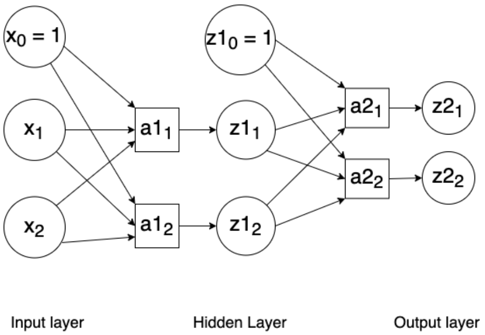
Forward Pass for a 2-layer Network

Calculate the values of $z1_j$ and $z2_k$ and E.

$$a1_j = \sum_i x_i W1_{ij} \qquad z1_j = g(a1_j) \tag{1}$$

$$a2_k = \sum_j z1_j W2_{jk} \qquad z2_k = g(a2_k) \tag{2}$$

$$E(z2, y) \tag{3}$$

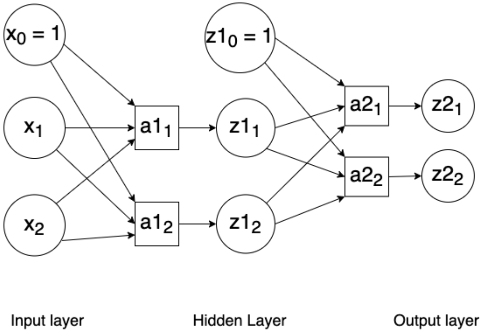


Backward Pass for a 2-layer Network

Calculate the gradients for $W1_{ij}$ and $W2_{jk}$.

$$\frac{\partial E}{\partial W2_{jk}} = \frac{\partial E}{\partial a2_k} z1_j = \delta 2_k z1_j, \quad \delta 2_k = \frac{\partial E}{\partial z2_k} g'(a2_k) \tag{4}$$

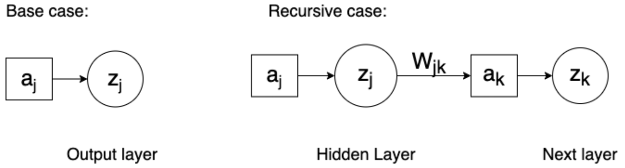
$$\frac{\partial E}{\partial W1_{ij}} = \frac{\partial E}{\partial a1_j} x_i = \delta 1_j x_i, \quad \delta 1_j = \left(\sum_k \delta 2_k W2_{jk} \right) g'(a1_j) \tag{5}$$



The recursive relationship

For unit j, $\delta_j = \dfrac{\partial E}{\partial a_j}$.

$$\delta_j = \begin{cases} \dfrac{\partial E}{\partial z_j} \times g'(a_j), & \text{base case, } j \text{ is an output unit} \\ \left(\sum_k \delta_k W_{jk}\right) \times g'(a_j), & \text{recursive case, } j \text{ is a hidden unit} \end{cases} \tag{6}$$



When should we use Neural Network?

- High dimensional or real-valued inputs, noisy (sensor) data.
- Form of target function is unknown (no model).
- Not important for humans to explain the learned function.

When should we NOT use Neural Network?

- Difficult to determine the network structure (number of layers, number of neurons).
- Difficult to interpret weights, especially in multi-layered networks.
- Tendency to over-fit in practice (poor predictions outside of the range of values it was trained on).

Decision Tree v.s. Neural Network

- Data types.
- Size of data set.
- Form of target function.
- The architecture.
- Interpret the learned function.
- Time available for training and classification.

Revisiting the Learning Goals

By the end of the lecture, you should be able to

- Explain the steps of the gradient descent algorithm.
- Explain how we can modify gradient descent to speed up learning and ensure convergence.
- Describe the back-propagation algorithm including the forward and backward passes.
- Compute the gradient for a weight in a multi-layer feed-forward neural network.
- Describe situations in which it is appropriate to use a neural network or a decision tree.