# CS188 Fall 2017 Section 12: Perceptrons / Neural Networks

# 1 Perceptron

We would like to use a perceptron to train a classifier with 2 features per point and labels +1 or −1. Consider the following labeled training data:

| Features $(x_1, x_2)$ | Label $y^*$ |
|---|---|
| $(-1, 2)$ | 1 |
| $(3, -1)$ | -1 |
| $(1, 2)$ | -1 |
| $(3, 1)$ | 1 |

1. Our two perceptron weights have been initialized to $w_1 = 2$ and $w_2 = -2$. After processing the first point with the perceptron algorithm, what will be the updated values for these weights?

2. After how many steps will the perceptron algorithm converge? Write "never" if it will never converge.

   Note: one step means processing one point. Points are processed in order and then repeated, until convergence.

# Perceptron → Neural Nets

Instead of the standard perceptron algorithm, we decide to treat the perceptron as a single node neural network and update the weights using gradient descent on the loss function.

The loss function for one data point is $Loss(y, y^*) = \frac{1}{2}(y - y^*)^2$, where $y^*$ is the training label for a given point and $y$ is the output of our single node network for that point. We will compute a score $z = w_1 x_1 + w_2 x_2$, and then predict the output using an activation function $g$: $y = g(z)$.

1. Given a general activation function $g(z)$ and its derivative $g'(z)$, what is the derivative of the loss function with respect to $w_1$ in terms of $g, g', y^*, x_1, x_2, w_1$, and $w_2$?

   $\dfrac{\partial Loss}{\partial w_1} =$

2. For this question, the specific activation function that we will use is

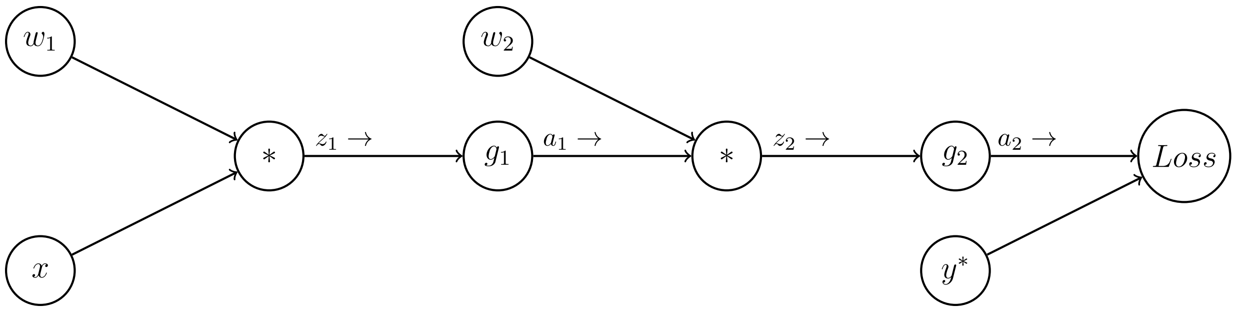
   $$g(z) = 1 \text{ if } z \geq 0 \text{ , or } -1 \text{ if } z < 0$$

   Given the gradient descent equation $w_i \leftarrow w_i - \alpha \frac{\partial Loss}{\partial w_1}$, update the weights for a single data point. With initial weights of $w_1 = 2$ and $w_2 = -2$, what are the updated weights after processing the first point?

3. What is the most critical problem with this gradient descent training process with that activation function?

# 2 Neural Nets

Consider the following computation graph for a simple neural network for binary classification. Here $x$ is a single real-valued input feature with an associated class $y^*$ (0 or 1). There are two weight parameters $w_1$ and $w_2$, and non-linearity functions $g_1$ and $g_2$ (to be defined later, below). The network will output a value $a_2$ between 0 and 1, representing the probability of being in class 1. We will be using a loss function $Loss$ (to be defined later, below), to compare the prediction $a_2$ with the true class $y^*$.




1. Perform the forward pass on this network, writing the output values for each node $z_1, a_1, z_2$ and $a_2$ in terms of the node's input values:

2. Compute the loss $Loss(a_2, y^*)$ in terms of the input $x$, weights $w_i$, and activation functions $g_i$:

3. Now we will work through parts of the backward pass, incrementally. Use the chain rule to derive $\frac{\partial Loss}{\partial w_2}$. Write your expression as a product of partial derivatives at each node: i.e. the partial derivative of the node's output with respect to its inputs. (Hint: the series of expressions you wrote in part 1 will be helpful; you may use any of those variables.)

4. Suppose the loss function is quadratic, $Loss(a_2, y^*) = \frac{1}{2}(a_2 - y^*)^2$, and $g_1$ and $g_2$ are both sigmoid functions $g(z) = \frac{1}{1+e^{-z}}$ (note: it's typically better to use a different type of loss, *cross-entropy*, for classification problems, but we'll use this to make the math easier).

   Using the chain rule from Part 3, and the fact that $\frac{\partial g(z)}{\partial z} = g(z)(1 - g(z))$ for the sigmoid function, write $\frac{\partial Loss}{\partial w_2}$ in terms of the values from the forward pass, $y^*$, $a_1$, and $a_2$:

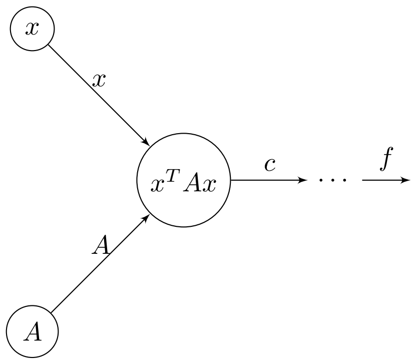
5. Now use the chain rule to derive $\frac{\partial Loss}{\partial w_1}$ as a product of partial derivatives at each node used in the chain rule:

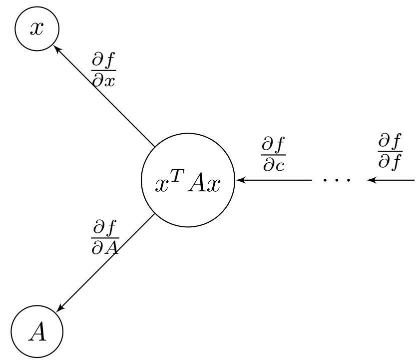
6. Finally, write $\frac{\partial Loss}{\partial w_1}$ in terms of $x, y^*, w_i, a_i, z_i$:

7. What is the gradient descent update for $w_1$ with step-size $\alpha$ in terms of the values computed above?

# 3 Vectorized Gradients

Let's compute the backward step for a node that computes $x^T Ax$, where $x$ is a vector with $m$ values, and $A$ is a matrix with shape $m \times m$. Thus, $c = \sum_{i=1}^{m} x_i \sum_{j=1}^{m} A_{ij} x_j = \sum_{i=1}^{m} \sum_{j=1}^{m} A_{ij} x_i x_j = \sum_{j=1}^{m} x_j \sum_{i=1}^{m} A_{ij} x_i$.




1. What is $\frac{\partial f}{\partial A_{ij}}$?

2. What is $\frac{\partial f}{\partial A}$?

3. What is $\frac{\partial f}{\partial x_k}$?

4. What is $\frac{\partial f}{\partial x}$?