# 1 Key Concepts

## 1.1 Logistic Regression

The simplest formulation of a neural network is logistic regression.

**Problem.** Given $m$ training examples with $n_x$ features each, we want to classify each test example as 0 or 1. Therefore, $X$ is shape $(n_x, m)$ and $y$ is shape $(1, m)$. Given some $x$, we want to calculate $\hat{y} = P(y = 1|x)$.

**Solution.** Create a weight matrix $w \in \mathbb{R}^{n_x}$ and $b \in \mathbb{R}$. We can calculate $z = w^T x + b$, and $\hat{y} = \sigma(w^T x + b)$, where $\sigma(z) = \frac{1}{1+e^{-z}}$.

Now, we define our loss function, applied to a single example, as

$$L(\hat{y}, y) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})) \ . \tag{1}$$

Observe that when $y = 0$, small $\hat{y}$ minimizes the loss, and when $y = 1$, large $\hat{y}$ minimizes the loss. Over the entire training set, we have the cost function

$$J(w, b) = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) \tag{2}$$

We then use gradient descent to find the optimal values for $w$ and $b$. For a single $w$ parameter, for example, we update with

$$w := w - \alpha \frac{dJ(w)}{dw} \ .$$

We can intuitively think about how this works. If some $w$ is too large, then $\frac{dJ(w)}{dw}$ will be positive, since $w$ is greater than the optimum $w$ and $J(w)$ is greater than the optimum $J$, so the slope is positive. Therefore, when we update with $w := w - \alpha \frac{dJ(w)}{dw}$, the new $w$ will be smaller and closer to optimum $w$.

**Q & A**.

1. Why don't we use the squared error loss function $L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$ ? It turns out that squared loss forms a non-convex optimization problem with multiple local optima. Cross-entropy loss, on the other hand, is a convex optimization function.