TRAINING NEURAL NETWORKS

Recall logistic regression:
1. Define prediction function $f_{\vec{w},b}$
2. Specify loss and cost fn. $J(\vec{w},b)$
3. Train on data by minimizing cost (using e.g. gradient descent)

For NN w/ TF: (w/ activation fn)
1. Build Sequential model of Dense layers
2. Specify loss fn to model.compile
3. Train on data w/ model.fit(X, Y, #epochs)

Activation Functions

So far we've built a NN by stringing together a bunch of log. reg. models (since activation fn = sigmoid). But we can allow a larger range of activation values w/ diff. fns, e.g. "ReLU" where $g(z) = \max(0, z)$. Linear also sometimes used (aka "none"), softmax.
(And others exist too.)

How to choose?
- Output layer:
  - for binary classification, Sigmoid is natural (b/c network learns to predict probabilities)
  - for regression, linear is natural, since this permits full $\mathbb{R}$ output (which is natural for regression)
  - for multiple classes, ReLU is natural: nonzero but multiple values.