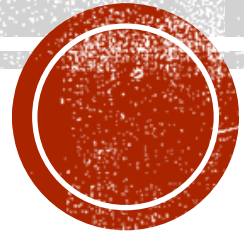
NNS WITH MULTIPLE OUTPUTS

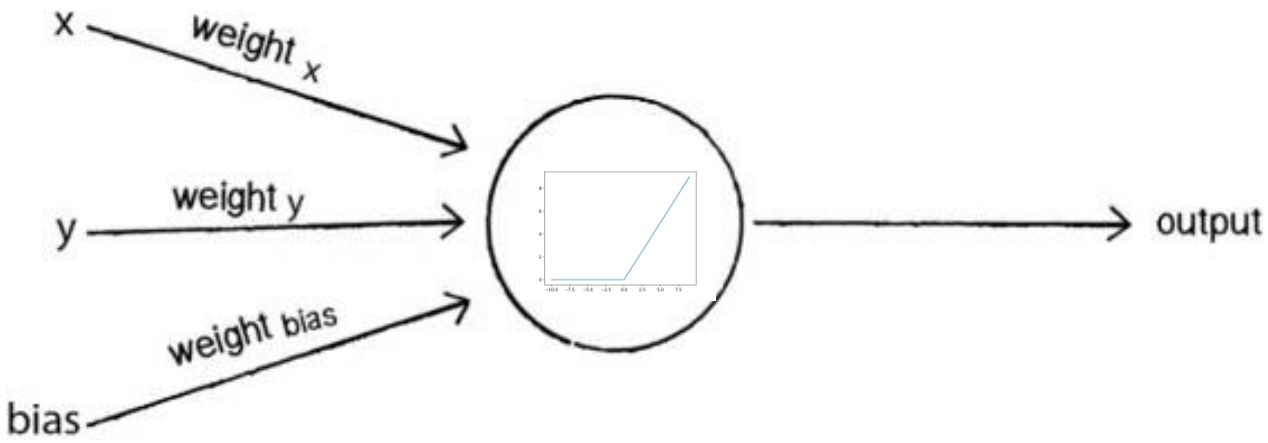
Dr. Brian Mc Ginley

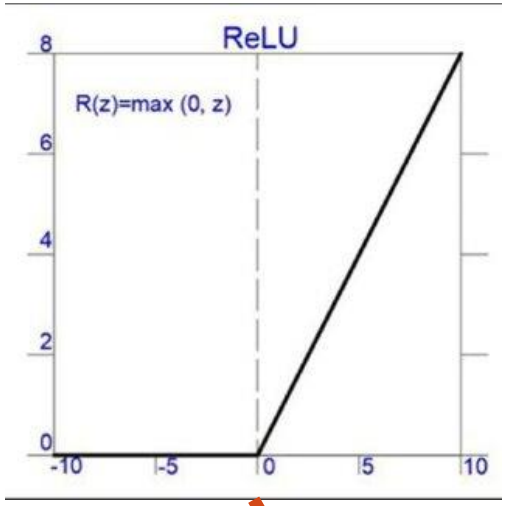


MORE THAN TWO CLASSES

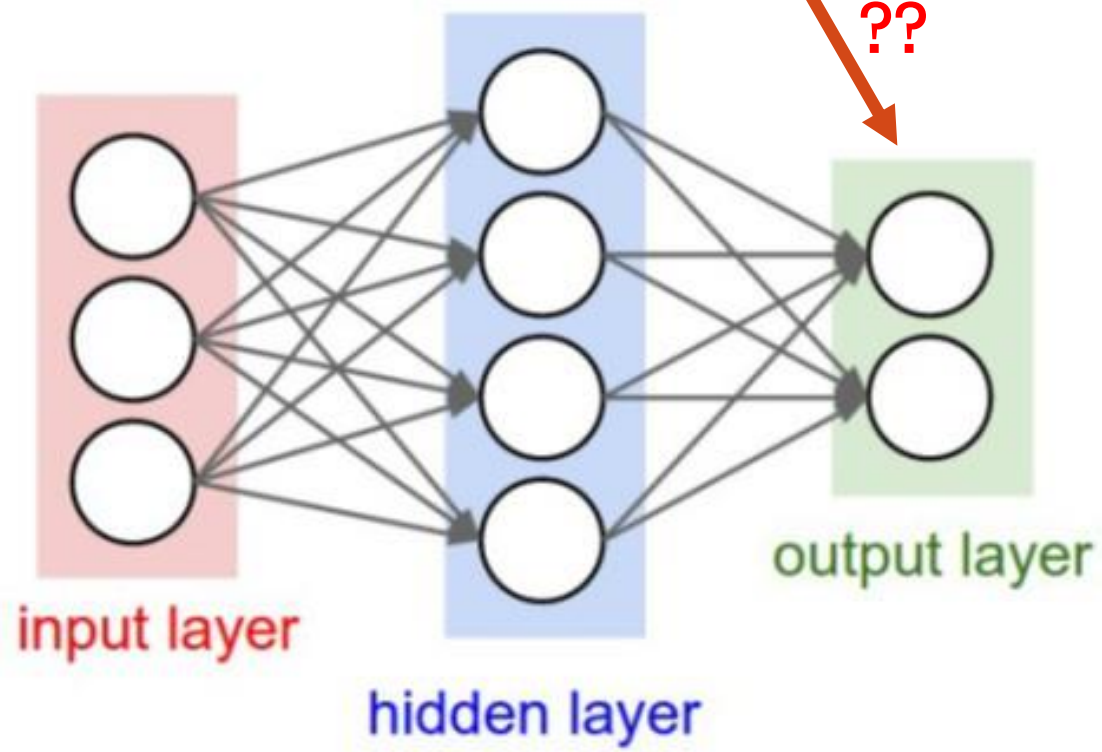
- We've seen binary classification.
  - E.g. survived in the Titanic dataset.
- What about multi-class problems?
  - Apple, banana, car, walk sign, zebra, zoo
  - Red, orange, yellow, green, blue, indigo, violet
  - Animal, plant, mineral

NEURAL NETWORK OUTPUTS





??



TERMINOLOGY

- Remember: With Logistic Regression, we passed the linear combination of weights and features (which potentially ranged from -∞ to +∞) through the Logistic/Sigmoid function to convert it into a range of probabilities (p) from 0 to 1.

$$f(x)\ (or\ p) = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + \cdots + w_n x_n)}}$$

- The Logit function did the opposite. It turned the probability (p) into the log of the odds (-∞ to +∞)

$$logit(p) = ln\left(\frac{p}{1-p}\right)$$

- Terminology:
  - In ML, a logit (as well as referring to the function above) also refers to the raw predictions which come out of the output layer of an NN (which potentially ranged from -∞ to +∞).

NEURAL NETWORK OUTPUTS

- If the output layer has a Relu activation function, the outputs are unbounded.
- Like with logistic regression, it would be much more useful if the outputs were in the range of 0-1
- With multiple outputs (instead of using Logistic/Sigmoid function – which is useful for binary classification), we use Softmax
- Softmax does some nice things:
  - Output range for each class is 0-1
  - Sum of output probabilities = 1

$$Softmax_i(NNOutputs) = \frac{e^{NNOutput_i}}{\sum_{j=1}^{c} e^{NNOutput_j}}$$

SOFTMAX EXAMPLE

- Let's look at the Iris Dataset – Setosa, Versicolour and Virginica
- Imagine a NN has the following raw outputs ($\boldsymbol{o}$):
  - Setosa = 1
  - Versicolour = 3
  - Virginica = -2
- Then the Softmax outputs for each class would be as follows (let's just look at Setosa (index = 1) below).

$$Softmax_i(\boldsymbol{o}) = \frac{e^{o_i}}{\sum_{j=1}^{c} e^{o_j}}$$

$$Softmax_1(\boldsymbol{o}) = \frac{e^{o_1}}{e^{o_1} + e^{o_2} + e^{o_3}}$$

SOFTMAX EXAMPLE

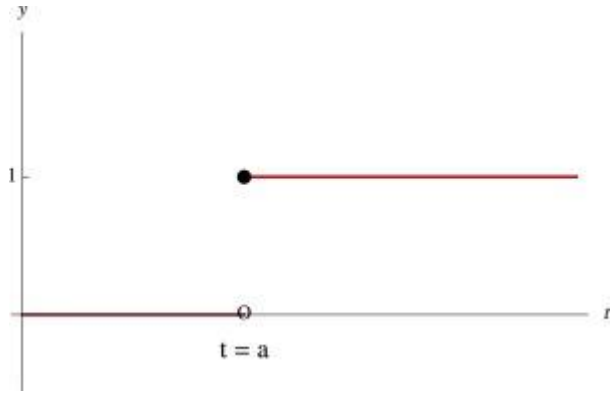
- Let's look at the Iris Dataset – Setosa, Versicolour and Virginica
- Imagine a NN has the following raw outputs ($\boldsymbol{o}$):
  - Setosa = 1
  - Versicolour = 3
  - Virginica = -2
- Then the Softmax outputs (rounded) for each class would be as follows
  - Not only getting a probability but a sum of 1

$$Softmax_1(\boldsymbol{o}) = \frac{e^{o_1}}{e^{o_1} + e^{o_2} + e^{o_3}} = \frac{e^{1}}{e^{1} + e^{3} + e^{-2}}$$
$$= \frac{2.7}{2.7 + 20.1 + 0.13} = 0.12$$

$$Softmax_2(\boldsymbol{o}) = 0.87$$

$$Softmax_3(\boldsymbol{o}) = 0.01$$

SOFTMAX VS ARGMAX



- Often with NNs, you see a function Argmax (instead of Softmax) being applied to the raw outputs.
  - Argmax basically returns the index of the maximum value in the array. I.e. it sets the highest output to 1 and all others to zero.
  - In the previous Iris flower example, the argmax outputs would have been:

Flower	Raw	Softmax	Argmax
Setosa	1	0.12	0
Versicolour	3	0.87	1
Virginica	-2	0.01	0

  - This is also useful but the big disadvantage is that the switch from 0-1 is a vertical step – i.e. there is no gradient – so no training
  - Common to use Softmax in training and Argmax in production.

SOFTMAX - TENSORFLOW

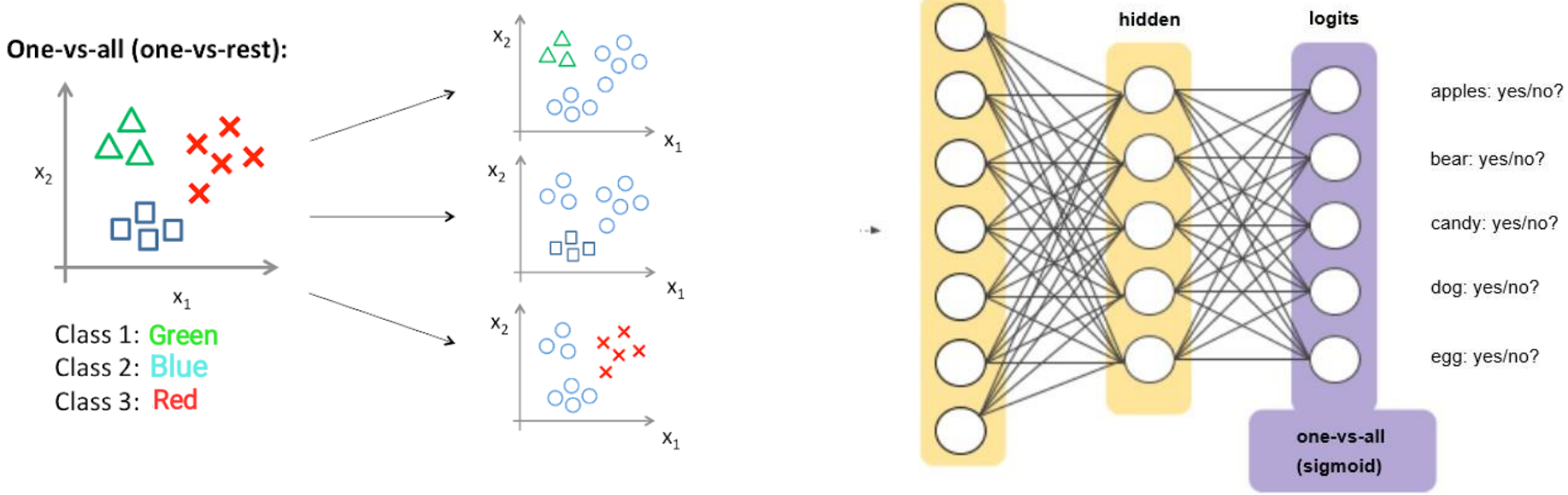
- From Tensorflow Documentation
  - Note: It is possible to bake the tf.nn.softmax function into the activation function for the last layer of the network. While this can make the model output more directly interpretable, this approach is discouraged as it's impossible to provide an exact and numerically stable loss calculation for all models when using a softmax output.
- For this reason, it's better to add Softmax on at the end and not apply it as an activation function on the trained model. This is also why "(from_logits=True)" is specifier for the loss function.
  - **If** you put Softmax on the end and fit using the model, make sure from_logits is False.

MULTI-CLASS

- Multi-Class, Single-Label Classification (One v One/Softmax):
  - An example may be a member of only one class.
  - Constraint that classes are mutually exclusive is helpful structure. Useful to encode this in the loss.
  - Use Softmax loss for all possible classes.
- Multi-Class, Multi-Label Classification (One v All):
  - An example may be a member of more than one class. (e.g. image contains apple and a banana or movie fits in multiple genres)
  - No additional constraints on class membership to exploit.
  - One logistic regression loss for each possible class.
- This is a harder problem which will require more investigation for you if you are interested! Candidate sampling is something to look up

ONE VS. ALL

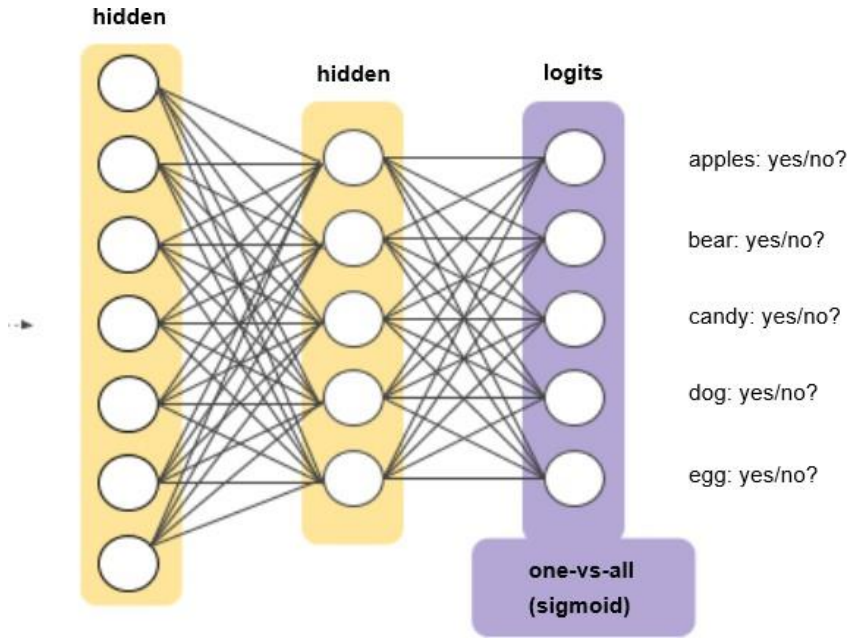
- One vs. all: provides a series of yes/no predictions for each label – i.e. N separate binary classifiers (computationally expensive)
  - Advantage is that it can be used for multi-label classification (where more than 1 of the outputs might be true) – e.g. in movie genres



- With Softmax, outputs are mutually exclusive (multi-class vs multi-label)

ONE-VS-ALL NN

- Create a unique output for each possible class
- Train that on a signal of "my class" vs "all other classes"
- Can do in a deep network, or with separate models
- So, one logistic regression output node in our model for every possible class.
- They share the internal representation through the rest of the model so these can be trained reasonably efficiently together.



ERROR METRICS

- How do we measure how well out multi-output NN is performing
- What have we seen so far?:
- MSE:

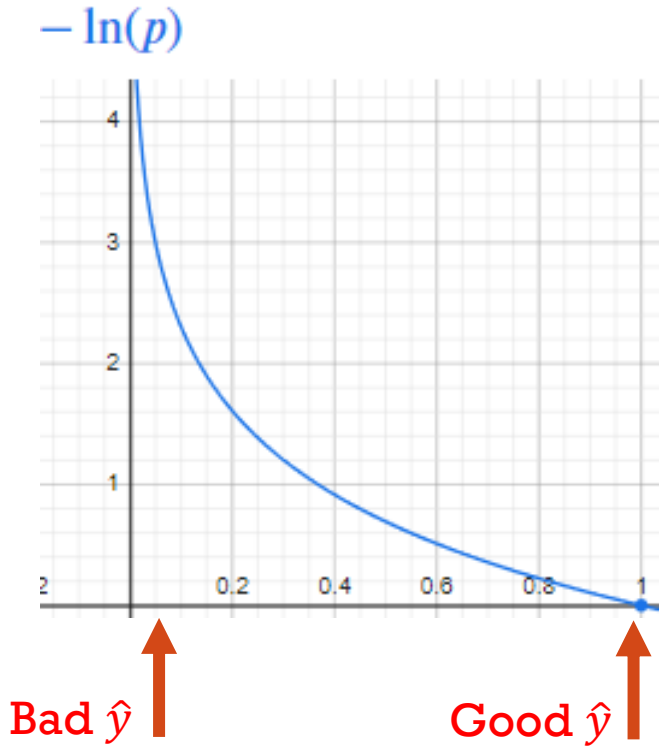
$$L = \frac{1}{2m}\sum_{i=0}^{m}(\hat{y}_i - y_i)^2$$

- Log Loss:

$$L(f_w(\boldsymbol{x}), y) = -\mathrm{y} * \log\big(f_w(\boldsymbol{x})\big) - (1-\mathrm{y})\log(1 - f_w(\boldsymbol{x}))$$

$$L(f_w(\boldsymbol{x}), y) = \begin{cases} -log\big(f_w(\boldsymbol{x})\big) & if\ y = 1 \\ -log\big(1 - f_w(\boldsymbol{x})\big) & if\ y = 0 \end{cases}$$

- Log Loss only works for 2 classes and MSE is better for regression



CROSS ENTROPY LOSS

- For multi-class NNs, we extend Log Loss

$$L = -\mathrm{y} * \log\big(f_{\boldsymbol{w}}(\boldsymbol{x})\big) - (1 - \mathrm{y})\log(1 - f_{\boldsymbol{w}}(\boldsymbol{x}))$$

- To this formula for Cross Entropy Loss:

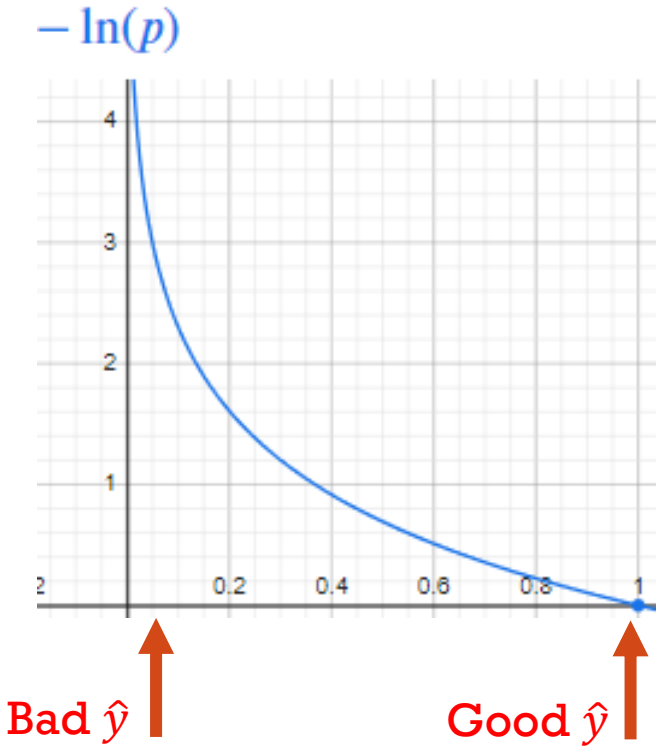
$$L = -log(CorrectOutputProbability)$$

- It's a simplified version of this formula

$$H(p, q) = -\sum_{x \in classes} p(x) log\ q(x)$$

True probability distribution (one-shot)

Your model's predicted probability distribution



CROSS ENTROPY LOSS

- Formula for Cross Entropy Loss: $L = -log(CorrectOutputProbability)$
- Back to our Iris data:
- If Versicolour is indeed correct,
  - $L = -log(0.87) = 0.139$
- If Setosa was correct answer
  - $L = -log(0.12) = 2.12$
- If Virginica was correct answer
  - $L = -log(0.01) = 4.6$
- As usual, remember to use natural log

Flower	Raw	Softmax	Argmax
Setosa	1	0.12	0
Versicolour	3	0.87	1
Virginica	-2	0.01	0