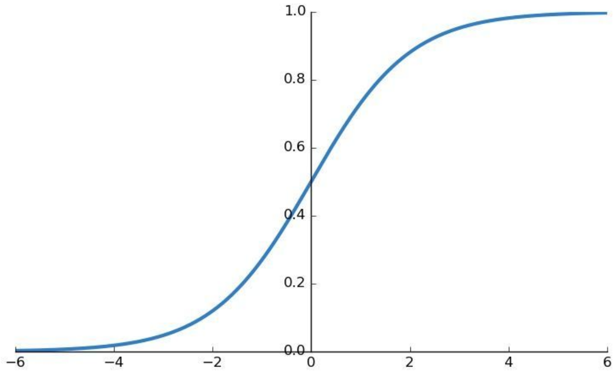
Losses and Activations

University of Victoria - PHYS 555

Classification

Binary classification recap

- $0 < \sigma < 1$
- Useful to squash layer output to represent binary probability
  $\rightarrow$ Bernoulli output distribution
- Expensive to compute
- Saturates at low and high input values $\rightarrow$ small slopes $\rightarrow$ low gradient signal $\rightarrow$ needs a log in the loss function to cancel the effect of the exp



$$\sigma(x) = \frac{1}{1 + \exp^{-x}}$$

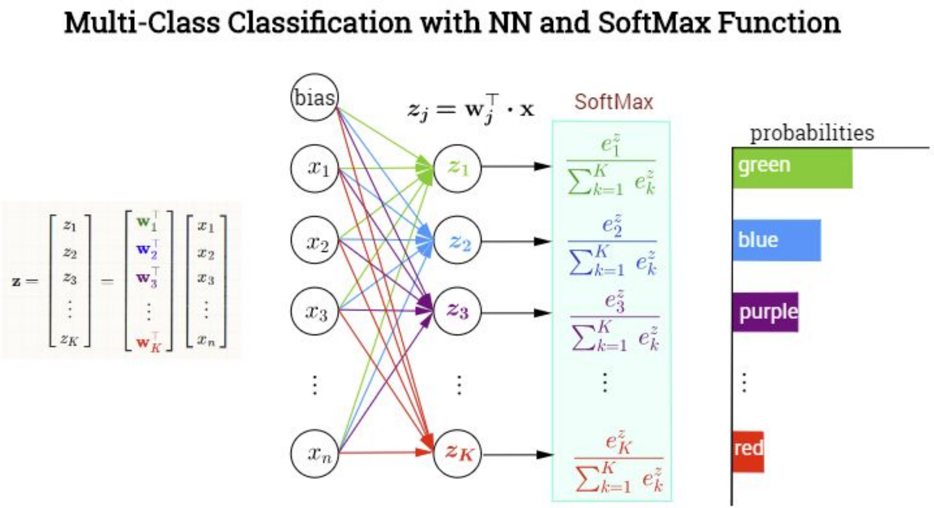
Softmax function

$$\text{softmax}(\mathbf{x}) = \frac{1}{\sum_{i=1}^{n} e^{x_i}} \cdot \begin{bmatrix} e^{x_1} \\ e^{x_2} \\ \vdots \\ e^{x_n} \end{bmatrix}$$

- vector of values in [0,1]
- behaves like a probability: denominator sums to 1
- is nicely differentiable
- the pre-activated function are called "logits"

Classification and softmax regression

- Multinoulli output distribution → multi-class output
- Produces a distribution over classes
- Predicted class is the one with the largest probability



Multi-class loss

- For a classifier with C classes, the network output is: $(f_1(\mathbf{x};\mathbf{w}),\ldots,f_C(\mathbf{x};\mathbf{w}))$
- We can interpret each output to be a probability, i.e. for NN: $f_j(\mathbf{x};\mathbf{w}) = \mathbf{x}^{\mathrm{T}}\mathbf{w}_j$

$$P(y=j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^{\mathsf{T}}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^{\mathsf{T}}\mathbf{w}_k}}$$

- A natural candidate for the loss is using negative log-likelihood:

$$nll = -\frac{1}{N}\sum_{n} \log P(y=j|\mathbf{x}_n)$$

Cross-entropy for multiclass

- Recall cross-entropy definition between 2 distributions ***p*** and ***q***:

$$H(p, q) = -\sum_x p(x) \log q(x)$$

- We can then write: $-\log P(Y = j|\mathbf{x}_n) = H(\delta(j), P(Y|\mathbf{x}_n))$

By minimizing our loss, we minimize the average cross-entropy between a deterministic posterior class probability and the estimated class probability

PyTorch Classification Losses Summary

`BCEWithLogitsLoss(ŷ, y)` = `BCELoss(sigmoid(ŷ), y)`

`CrossEntropyLoss(ŷ, y)` = `NLLLoss(log(softmax(ŷ)), y)`

Loss function	Usage	Example Using probabilities	Example Using logits
BCELoss or *BCEWithLogitsLoss*	Binary classification	*BCELoss* y_true: 1 y_pred: 0.8	*BCEWithLogitsLoss* y_true: 1 y_pred: 0.8
NLLLoss or *CrossEntropyLoss*	Multiclass classification	*NLLLoss* y_true: 2 y_pred: 0.30 0.15 0.55	*CrossEntropyLoss* y_true: 2 y_pred: 1.5 0.8 2.1

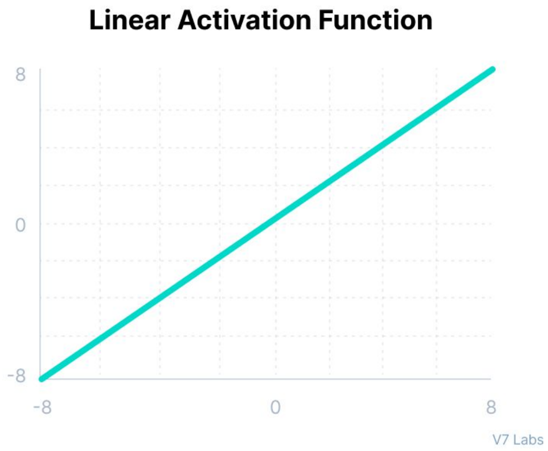
credits: Raschka Book

Regression

Activation in regression problems

- For continuous targets, we do not need an nonlinear function == Identity

- However beware of high dynamic range of vastly different values in your target variables.
  - Use a transformation if this is the case



MSE and MAE

- L2 loss or Mean Square Error: `MSELoss`

$$\texttt{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(\mathbf{x}_i, \theta))^2$$

- L1 loss or Mean Absolute Error `L1Loss`

$$\texttt{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - f(\mathbf{x}_i, \theta)|$$

```
More robust to outliers
slightly less stable
```

Quiz

1. How do you transform a one label supervised regression problem into a classification one?

2. How do you transform a binary classification into a regression problem?