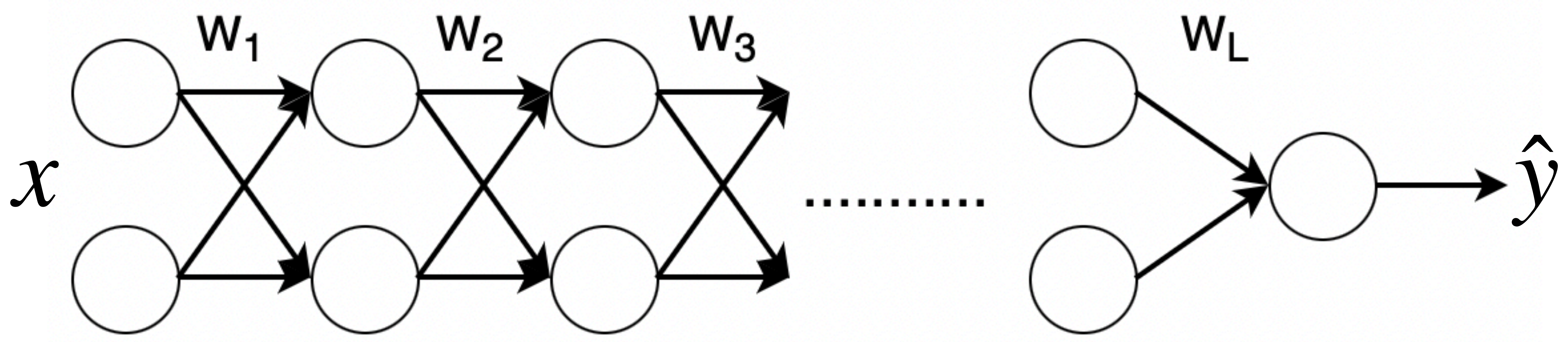
Activation Initialisation and Regularisation

Perumadura De Silva

Content

- Linear Activation and issues
- Tanh activation and issues
- RelU activation and its variants
- Weights initialisation methods
- Weights regularisation methods
    - Intuition
    - L1 and L2 regularisation
    - Dropout

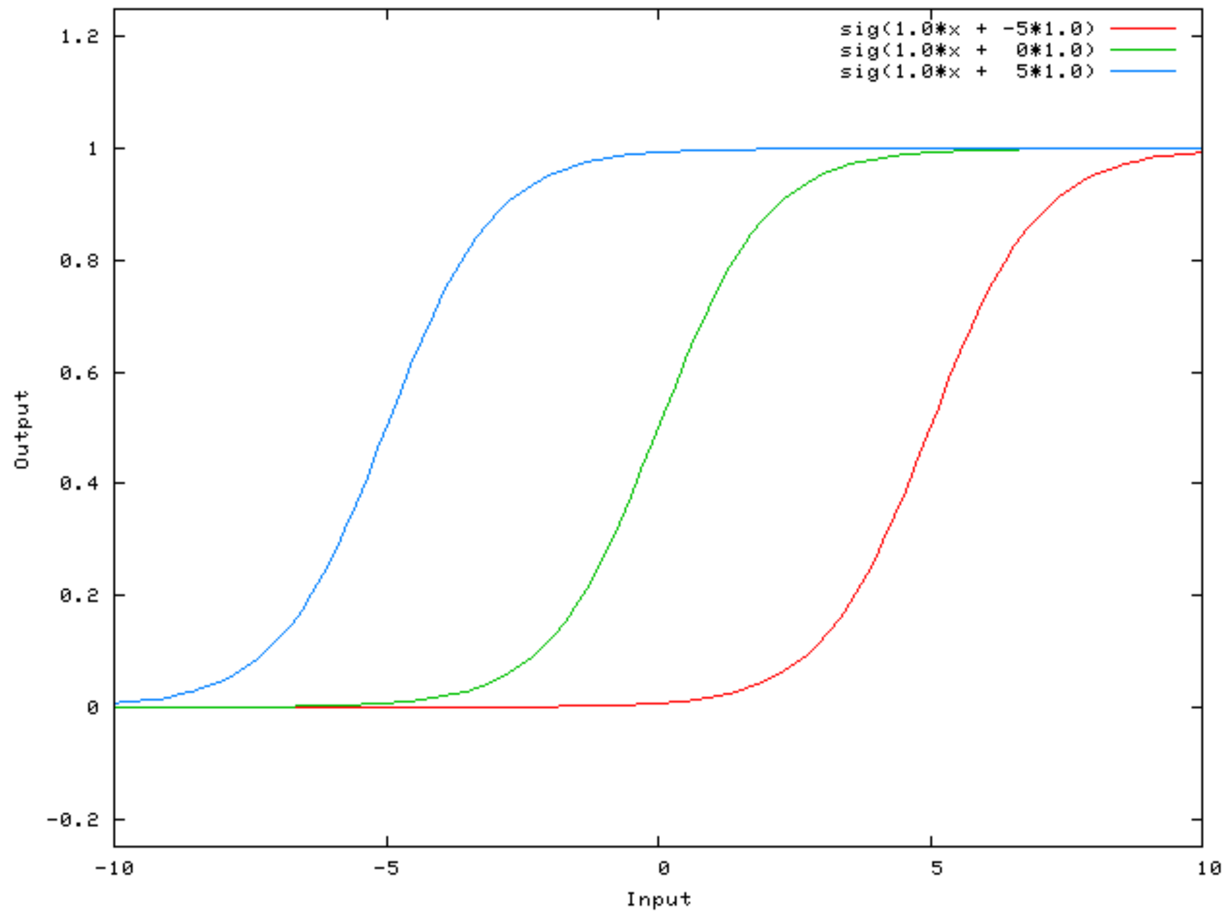
Linear activation and issues



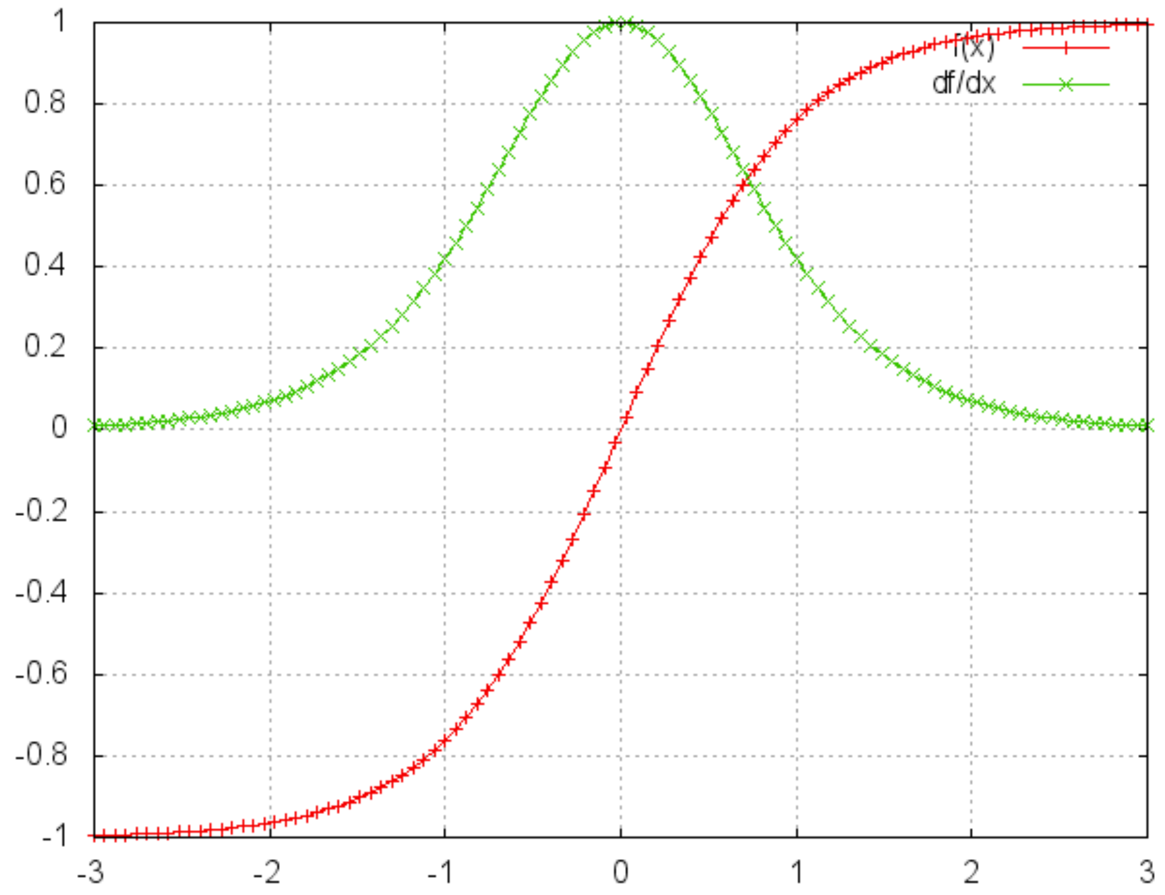
- $a_L = W_L W_{L-1} \dots W_3 W_2 W_1 x$ and $a_l = W_l z_{l-1} = W_l a_{l-1}$
- Where x: 2×1, $W_1 \dots W_{L-1}$: 2×2, W_L: $1x2$ and a_L: $1x1$
- Take $W^l = \begin{bmatrix} 1.5 & 0 \\ 0 & 1.5 \end{bmatrix}$ where $l = 1....(L-1)$
- $a_L = W_L 1.5^{L-1} x$; large outputs leads to larger gradients; Exploding gradient
- Take $W^l = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$ where $l = 1....(L-1)$
- $a_L = W_L 0.5^{L-1} x$; smaller outputs leads to smaller gradients; Vanishing gradient

Tanh activation and issues

- Tanh function: $f(z^l) = \dfrac{2}{1 + e^{-2z^l}} - 1$
- Derivative of tanh: $\dfrac{\partial f(z^l)}{\partial z^l} = 1 - tanh^2(z^l) = f^{'}(z^l) = \dfrac{\partial a^l}{\partial z^l}$; where $z^l = W^l a^{l-1}$ and $a^{l-1} = f(z^{l-1})$; l is the layer
- When inputs z^l are close to “0”; $tanh(z^l)^2 \approx 0$; $f^{'}(z^l)$ is a strong gradient
- When inputs z^l are very large; $tanh(z^l)^2 \approx 1$; $f^{'}(z^l)$ is a weak gradient
- Take error gradient $\dfrac{\partial L}{\partial W_L} = \dfrac{\partial L}{\partial a_L} \dfrac{\partial a_L}{\partial z_L} \dfrac{\partial z_L}{\partial W_L}$
  - with a weak $\dfrac{\partial a_L}{\partial z_L}$, $\dfrac{\partial L}{\partial W_L}$ vanishes; vanishing gradient problem
- Going further back: $\dfrac{\partial L}{\partial W_{L-1}} = \dfrac{\partial L}{\partial a_L} \dfrac{\partial a_L}{\partial z_L} \dfrac{\partial z_L}{\partial a_{L-1}} \dfrac{\partial a_{L-1}}{\partial z_{L-1}} \dfrac{\partial z_{L-1}}{\partial W_{L-1}}$; $\dfrac{\partial z_L}{\partial a_{L-1}} = W_L$
  - Large weights could lead to large $\dfrac{\partial L}{\partial W_{L-1}}$ gradients; Exploding gradient problem



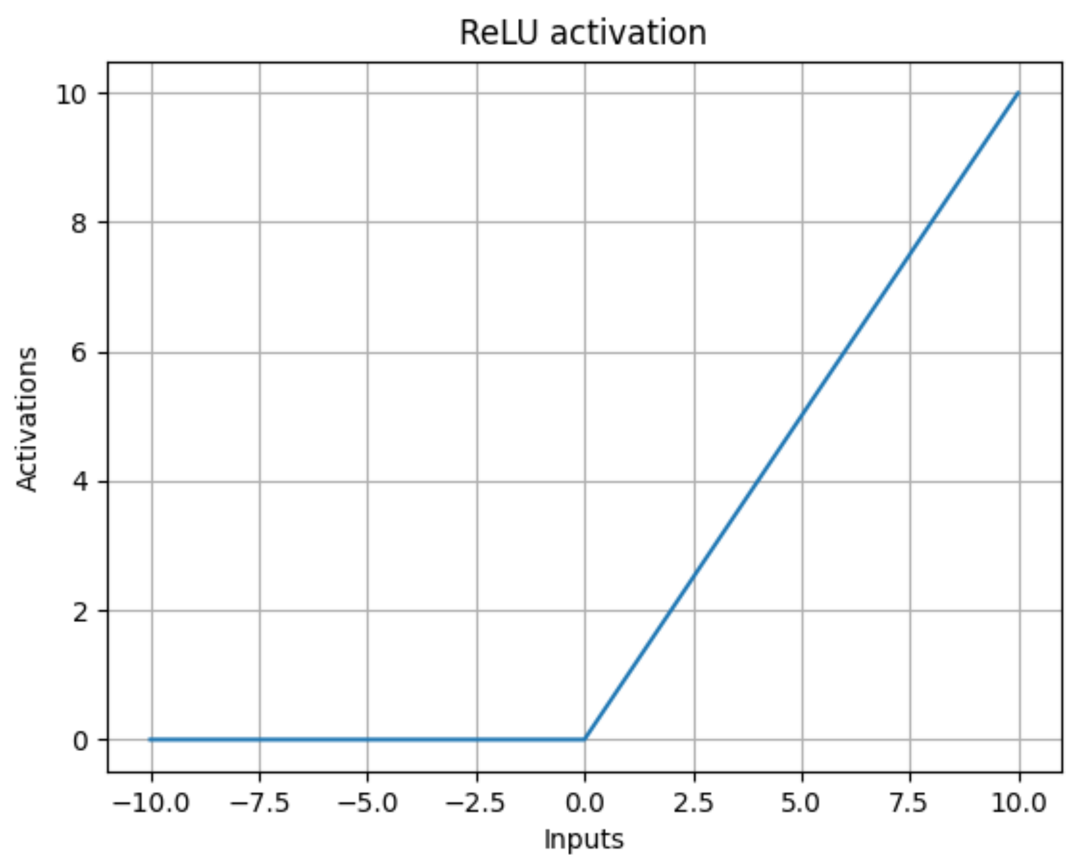
Ref-0: Sigmoid with different biases

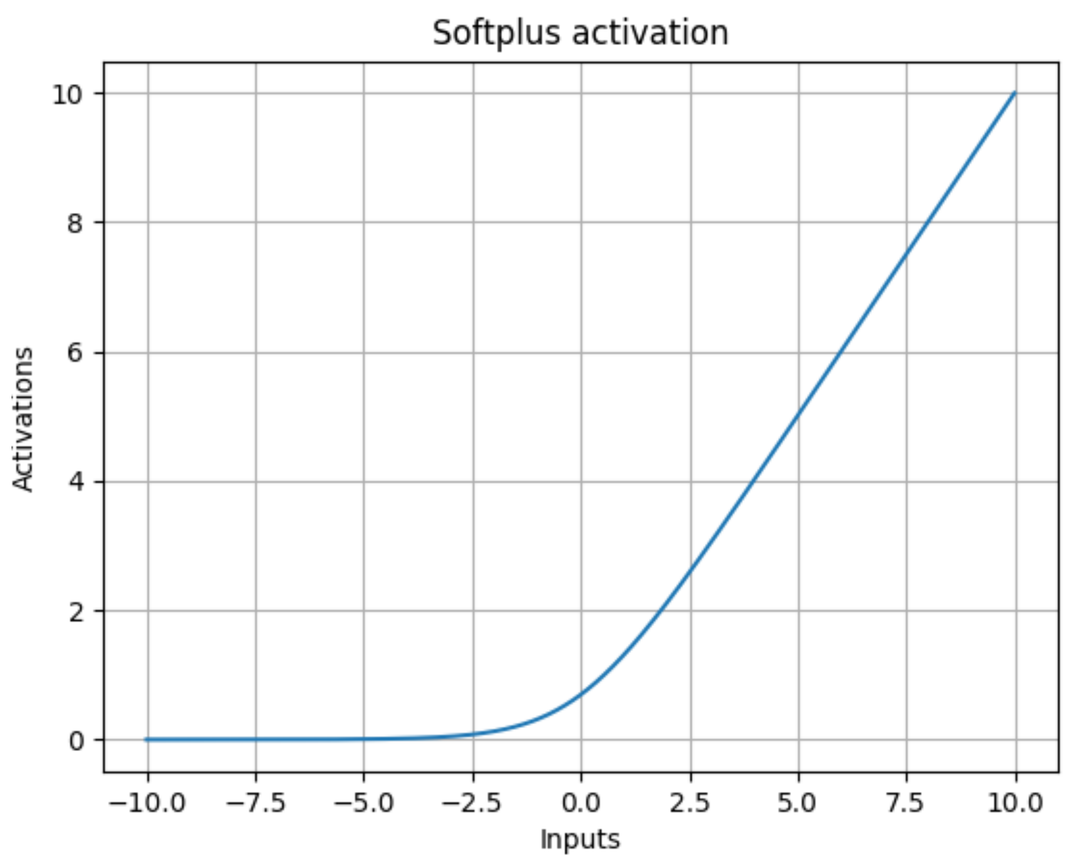


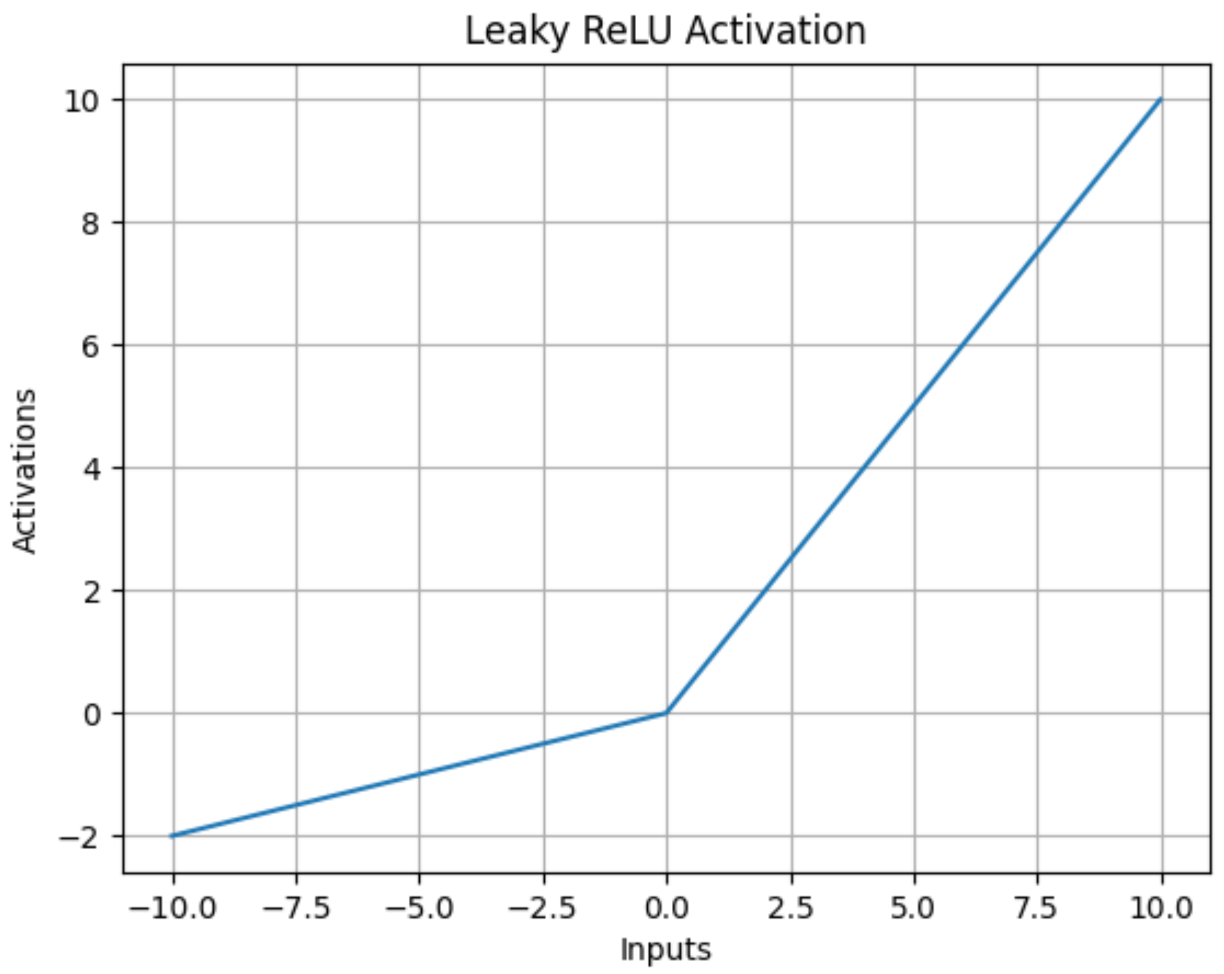
tanh(x) function against $\dfrac{df}{dx}$

ReLU activation and its variants

- ReLU activation: $f(z^l) = max(0, z^l)$
- Derivative of ReLU: $\frac{\partial f(z^l)}{\partial z^l} = 1 = f'(z^l)$
- When $z^l = 0$; $f'(z^l)$ is undefined. Since features are in floating points; this is not a problem most of the time
- ReLU variant Softplus addresses this issue:
  - $f_{softplus}(z^l) = log(1 + e^{z^l})$
- For $z^l < 0$; $f(z^l) = 0$ and $f'(z^l) = 0$. Gradient may not recover; Stationary weights
- Leaky ReLU addresses this issue by allowing weak negative outputs
  - $f_{leReLU}(z^l) = \begin{cases} \gamma z^l & z^l < 0 \\ z^l & z^l \geqslant 0 \end{cases}$ where γ is the negative slope; e.g $\gamma = 0.01$
- With large weights, it is possible to have an exploding gradient and vanishing gradient when weights are small

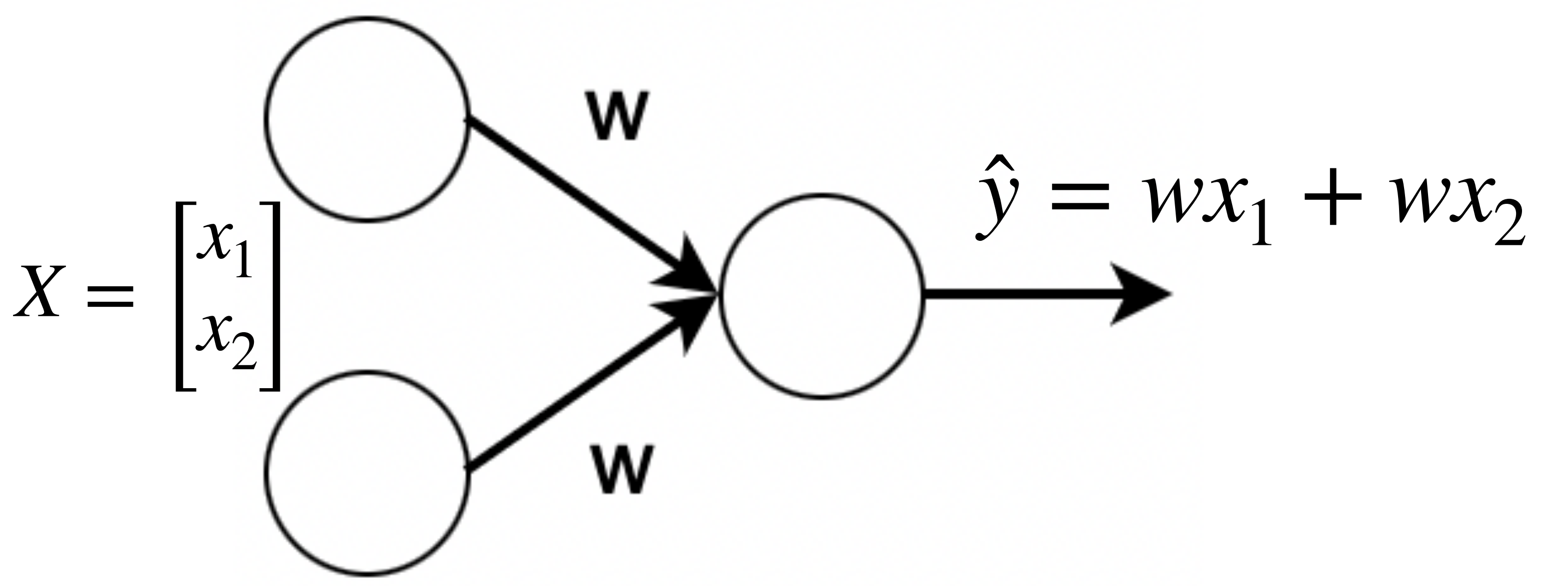


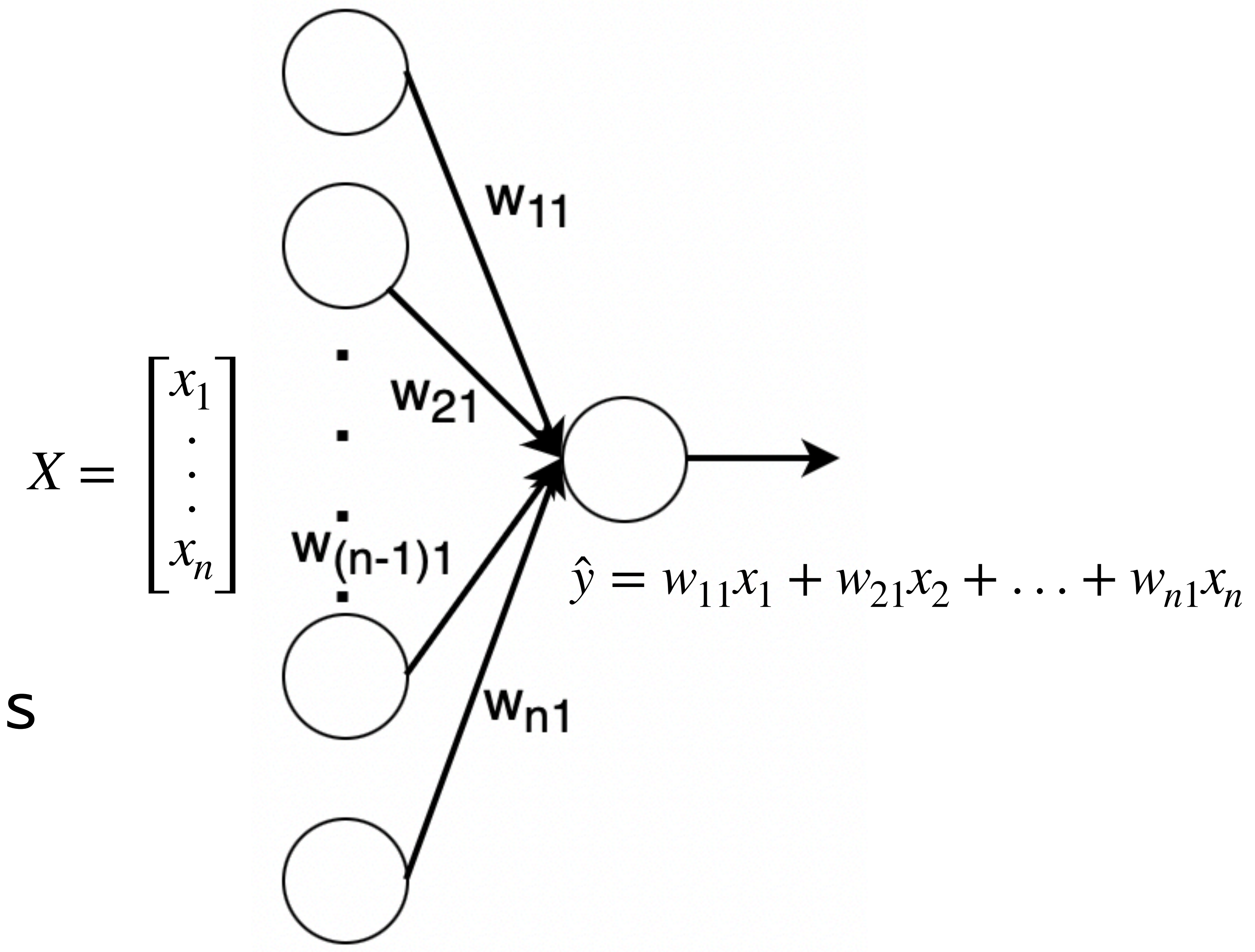




Weight initialisation methods

- Consider the neural network with two inputs and one outputs:
  - $\hat{y} = wx_1 + wx_2$
  - Weights contributes equally to the cost, so they will never differ from each other -> Symmetry breaking problem
  - If $w = 0$ no update at all
- If weights are randomly initialised to large values: Exploding gradient
- So the weights must be initialised in such a way that:
  - Weights are distributed around 0 (has 0 mean);
    - tanh has the best gradients when $z^l \approx 0$
    - ReLU needs small weights to prevent exploding gradients
- So a better initialisation method must address the above conditions



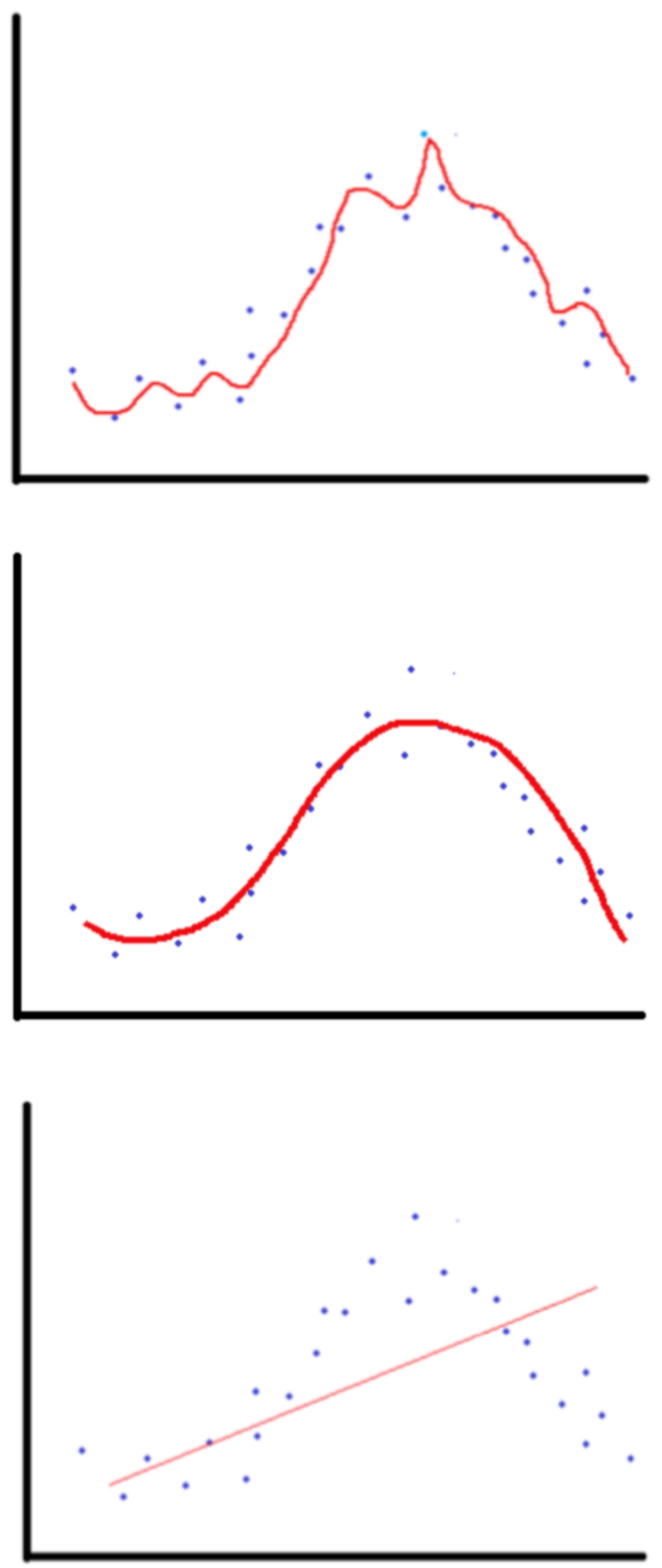


Weight initialisation methods

- Assume for a layer l:
- Weights and activations are independent and identically distributed
- Weight distribution and activation distribution are independent from each other
- Weights distribution has a 0 mean; $mean(W^l) = 0$
- For tanh activation: Xavier/Glorot Initialisation:
  - $mean(a^l) \approx 0$ and $Var(a^1) = \ldots. = Var(a^l) = Var(a^{l+1}) \ldots.. = Var(a^L)$; to satisfy a good gradient signal
  - $Var(a^l) = Var(z^l)$
  - Weight initialisation: $W^l = \mathcal{N}\left(\mu = 0, \sigma = \sqrt{\dfrac{2}{n^{l-1} + n^l}}\right)$ or $W^l = U(-limit, limit)$ where $limit = \sqrt{\dfrac{6}{n^{l-1} + n^l}}$
- For ReLU activation: He Initialisation:
  - $mean(a^l) \approx 0$ will not hold since ReLU operates between 0 and x
  - Weight initialisation: $W^l = \mathcal{N}\left(\mu = 0, \sigma = \sqrt{\dfrac{2}{n^{l-1}}}\right)$ or $W^l = U(-limit, limit)$ where $limit = \sqrt{\dfrac{6}{n^{l-1}}}$

Weights regularisation methods: Intuition

- When the dataset is too simple compared to the model:
  - The model has a high variance and low bias
  - The model over fits to data by memorising the datapoints
- So now the idea is to reduce the variance and increase bias
- Regularisation achieves this by reducing the model complexity/ capacity.
- Model complexity/capacity reduces when the model parameters shrink towards zero
- Take the following two models:
  - $\hat{y}_1 = \theta_0 + \theta_1 x + \theta_2 x^2$ (appropriate capacity) and $\hat{y}_2 = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_3 x^4$ (high capacity)
  - We add the regularisation term to the cost function of $\hat{y}_2$:
    - $C(\theta) = L(\theta, x, y) + \lambda\theta_3 + \lambda\theta_4$; where $\lambda\theta_3 + \lambda\theta_4$ is the regularisation
    - When $C(\theta) -> 0$, $\theta_3, \theta_4 -> 0$ so $\hat{y}_1 \approx \hat{y}_2$; Simplifies the hypothesis



High variance
Low bias
Overfitting

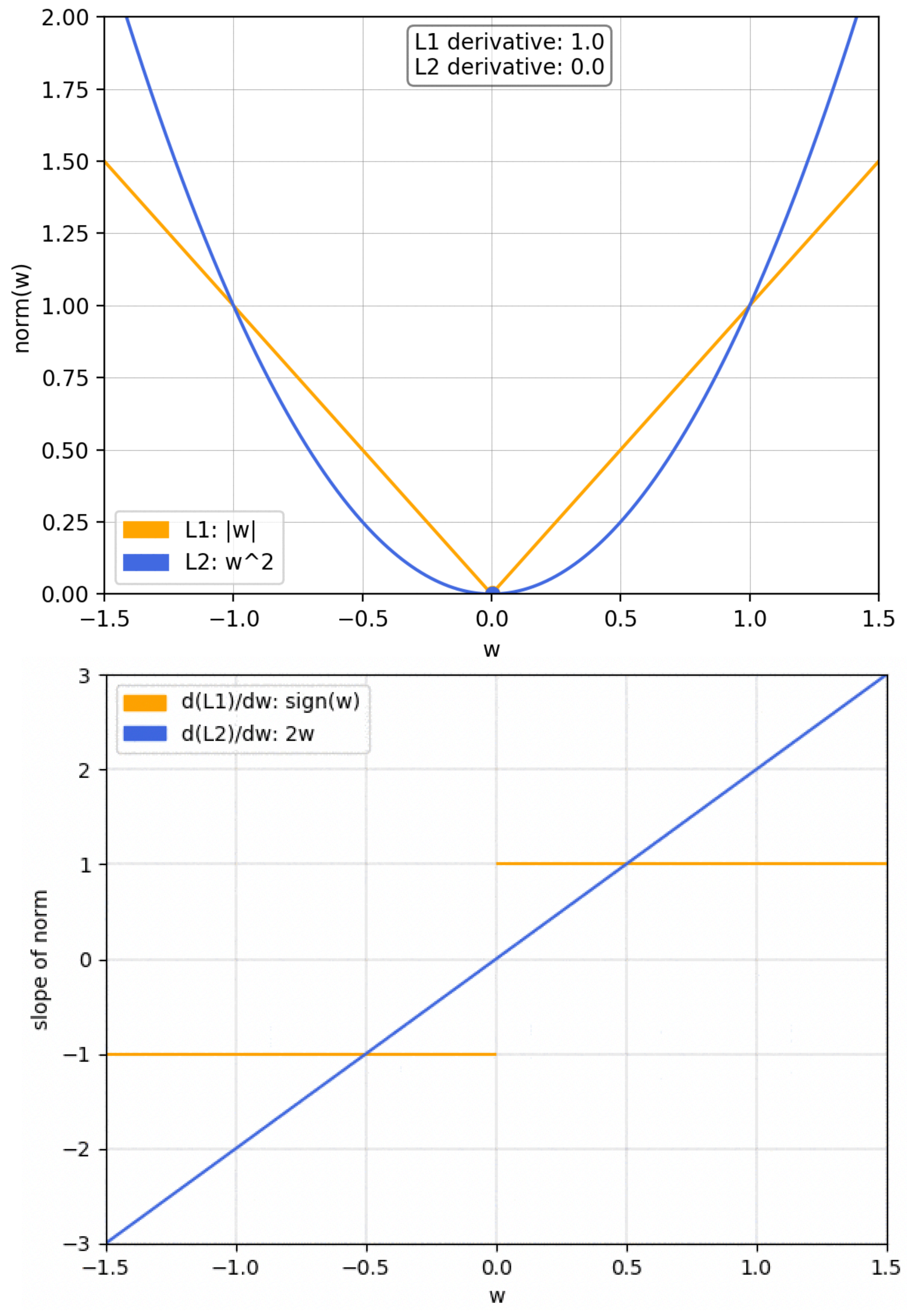
Appropriate variance
Appropriate bias
Good generalisation

Low variance
High bias
Underfitting

Images: Ref-1

Weights regularisation methods: L1 and L2

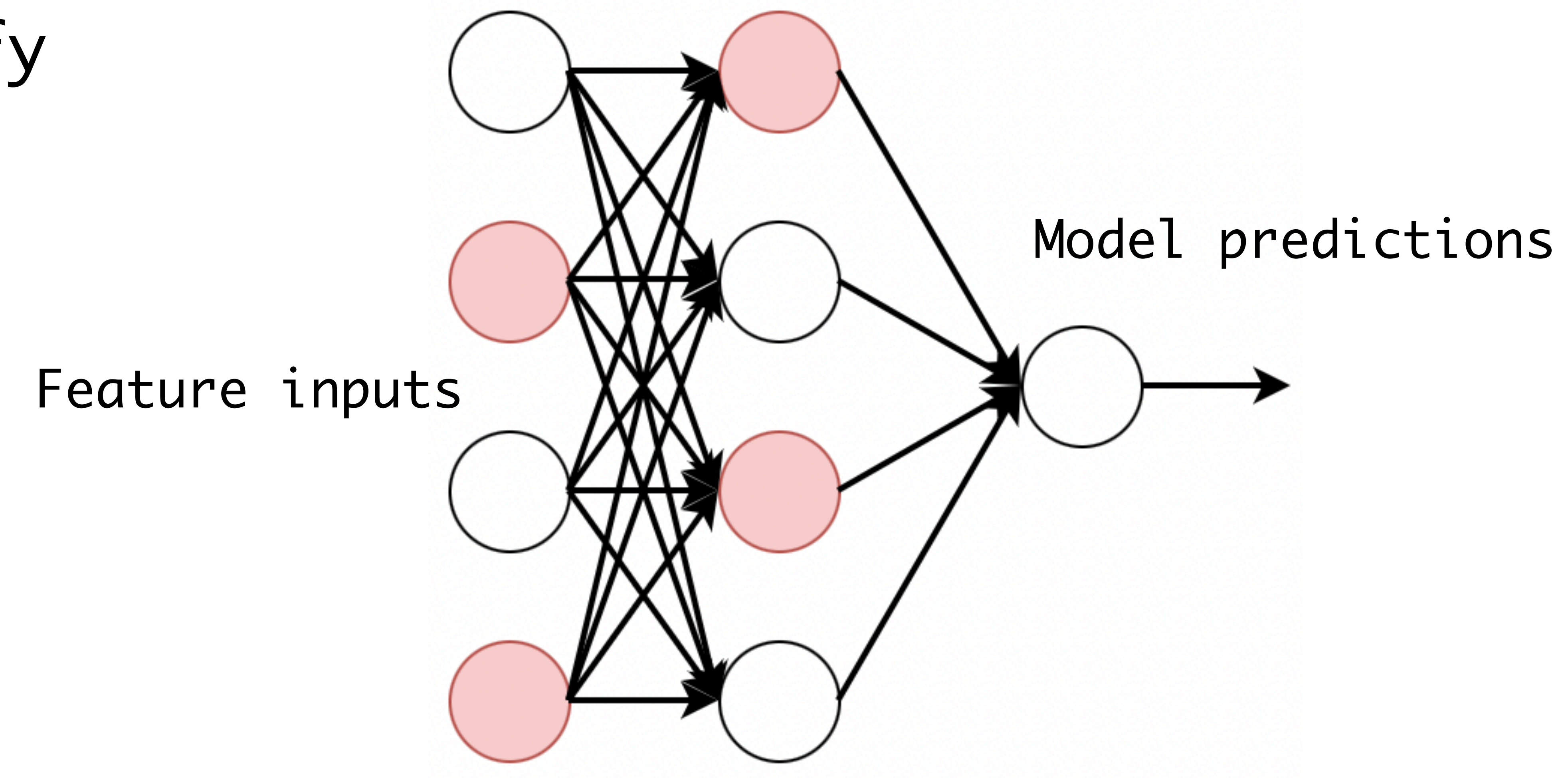
- A neural network can have a large number of parameters; We don't know exactly which parameter to regularise.
- Therefore we consider all the parameters in the regularisation term
- The modified cost function (L1):
  - $C(W,x,y) = L(W,x,y) + \lambda \sum_{l=1}^{L} ||W^l||_1$; where $||W^l||_1 = \sum_{i=1}^{N} |W_i^l|$; N is all the parameters in the layer l
  - Weight update at layer l: $W^l := W^l - \alpha \frac{\partial L}{\partial W^l} \pm \alpha\lambda$; α: learning rate
- The modified cost function (L2):
  - $C(W,x,y) = L(W,x,y) + \lambda \sum_{l=1}^{L} ||W^l||_2^2$; where $||W^l||_2^2 = \sum_{i=1}^{N} (W_i^l)^2$; N is all the parameters in the layer l
  - Weight update at layer l: $W^l := (1 - 2\alpha\lambda)W^l - \alpha \frac{\partial L}{\partial W^l}$



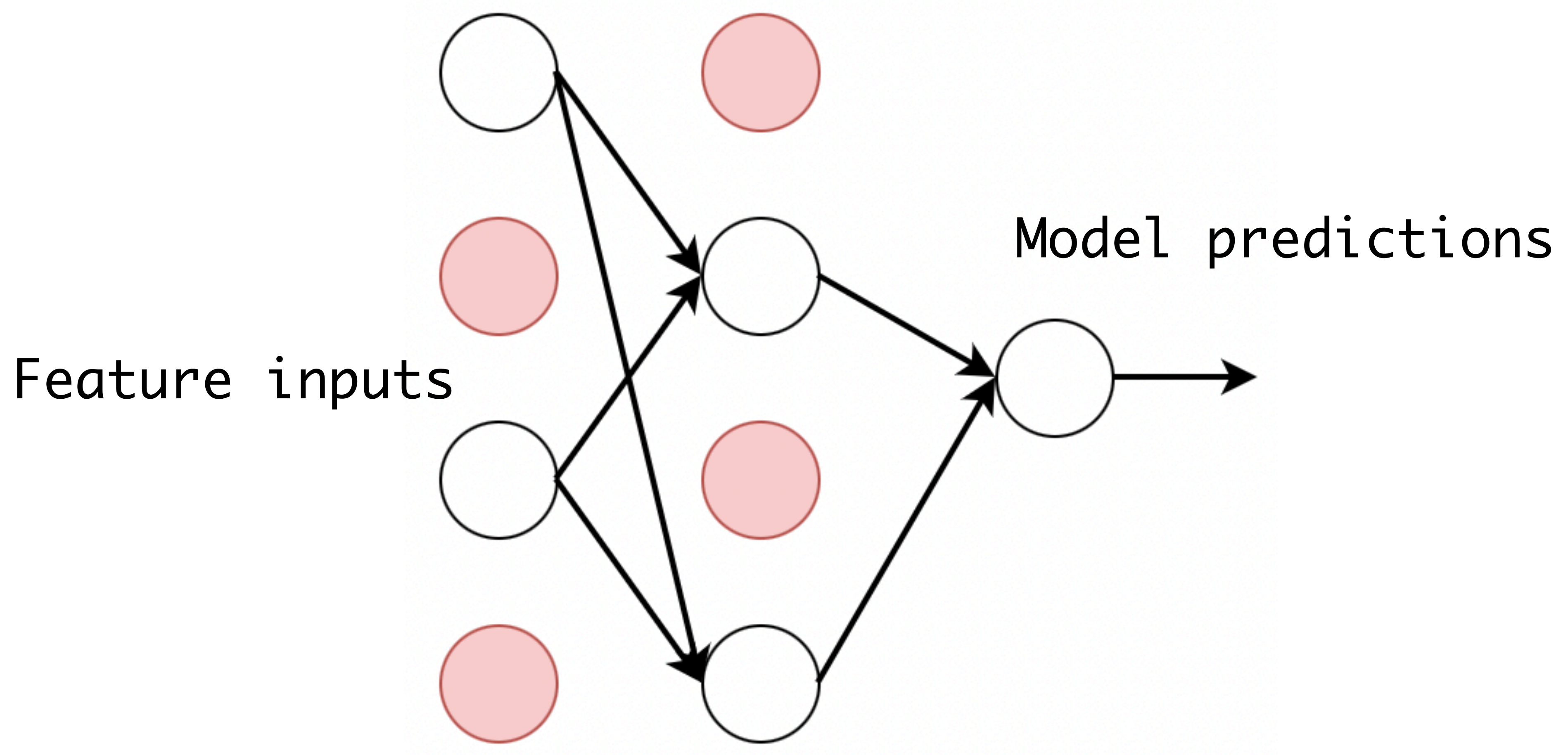
Ref-2: Top: Weight norm vs weight size for one parameter
Bottom: How the norm changes as the weight parameter changes

Weights regularisation methods: Dropout

- Similar to the case of L1 and L2 the idea is to simplify the model
  - For a given layer: remove (switch off) the neurones randomly with a given probability p at each training iteration (Applied for each batch)
  - The removal of the neurones regularises the layer by reducing the co-adaptation between neurones
  - Unlike the L1 and L2 case, no need to modify the loss
  - When trained with dropout: It is similar to performing an averaged prediction from multiple neural networks
  - At inference:
    - Since the neurones are dropped with a probability, at inference neurones express the features strongly
    - Therefore multiply the the activations by $1-p$ to reduce the strength



For a selected set of layers,
select neurones with a probability p



Switch off the connections of the selected neurones

References

- Ref-0: https://medium.com/deeper-learning/glossary-of-deep-learning-bias-cf49d9c895e2
- Ref-1: https://towardsdatascience.com/bias-and-variance-in-machine- learning-b8019a5a15bc
- Activation functions:
  - https://arxiv.org/pdf/2004.06632.pdf
  - ReLU: https://www.cs.toronto.edu/~fritz/absps/reluICML.pdf
- Weight initialisation:
  - https://www.deeplearning.ai/ai-notes/initialization/
  - Xavier/Glorot: https://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf
  - He: https://pouannes.github.io/blog/initialization/
- L1 and L2 Regularisation:
  - Ref-2: https://towardsdatascience.com/visualizing-regularization-and-the-l1-and-l2-norms-d962aa769932
  - https://www.youtube.com/watch?v=u73PU6Qwl1I
  - https://srdas.github.io/DLBook/ImprovingModelGeneralization.html#Regularization
  - https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/l2-regularization
- Dropout:
  - https://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf
  - https://programmathically.com/dropout-regularization-in-neural-networks-how-it-works-and-when-to-use-it/
  - https://harvard-iacs.github.io/2019-CS109B/a-sections/a-section1/notes/cs109b_asec1_notes_dropout.pdf