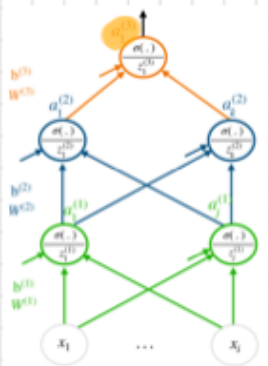
Lecture 21 - 14/11/24

- **Example ANN with several hidden layers:**



- It is for **binary classification**
- Its **loss function** is:

$$L(a^{(o)}, y) = -(y \cdot \log(a) + (1-y) \cdot \log(1-a))$$

- What we do now is:
  - **Back Propagation!**

$$L(a^{(3)}, y) = -(y \cdot \log(a_1^{(3)}) + (1-y) \cdot \log(1-a_1^{(3)}))$$

$$\boxed{\frac{\partial L}{\partial a_1^{(3)}}} = -\left[y \cdot \frac{1}{a_1^{(3)}} + (1-y) \cdot \left(\frac{1}{1-a_1^{(3)}} \cdot -1\right)\right]$$

$$= -\left[\frac{y}{a_1^{(3)}} + \frac{(1-y)}{1-a_1^{(3)}}\right] = -\frac{y(1-a_1^{(3)}) + a_1^{(3)}(-1+y)}{a_1^{(3)}(1-a_1^{(3)})}$$

$$= -\frac{y - \cancel{y a_1^{(3)}} - a_1^{(3)} + \cancel{a_1^{(3)} y}}{a_1^{(3)}(1-a_1^{(3)})} = \boxed{\frac{-y + a_1^{(3)}}{a_1^{(3)}(1-a_1^{(3)})}}$$

$a_1^{(3)} - y$

$a_k^{(2)}(a_1^{(3)} - y)$

$$\frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} = a_1^{(3)}(1 - a_1^{(3)}) \qquad \frac{\partial L}{\partial z_1^{(3)}} = \frac{\partial L}{\partial a_1^{(3)}} \cdot \frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} \qquad \frac{\partial L}{\partial w_{k,1}^{(3)}} = \frac{\partial L}{\partial z_1^{(3)}} \cdot a_k^{(2)}$$

$$\frac{\partial z_1^{(3)}}{\partial a_k^{(2)}} = w_{k,1}^{(3)} \qquad \frac{\partial L}{\partial b_1^{(3)}} = \frac{\partial L}{\partial z_1^{(3)}}$$

$$\frac{\partial L}{\partial a_k^{(2)}} = \frac{\partial L}{\partial z_1^{(3)}} \cdot w_{k,1}^{(3)}$$

$$\frac{\partial a_k^{(2)}}{\partial z_k^{(2)}} = a_k^{(2)}(1 - a_k^{(2)}) \qquad \frac{\partial L}{\partial z_k^{(2)}} = \frac{\partial L}{\partial a_k^{(2)}} \cdot \frac{\partial a_k^{(2)}}{\partial z_k^{(2)}} \qquad \frac{\partial L}{\partial w_{j,k}^{(2)}} = \frac{\partial L}{\partial z_k^{(2)}} \cdot a_j^{(1)}$$

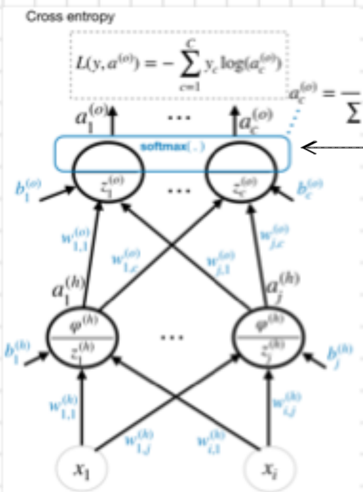
$$\frac{\partial z_k^{(2)}}{\partial a_j^{(1)}} = w_{j,k}^{(2)} \qquad \frac{\partial L}{\partial b_k^{(2)}} = \frac{\partial L}{\partial z_k^{(2)}}$$

$$\frac{\partial L}{\partial a_j^{(1)}} = ?$$

$$\frac{\partial a_j^{(1)}}{\partial z_j^{(1)}} = a_j^{(1)}(1 - a_j^{(1)})$$

- **Example ANN with Softmax output layer**

NOTE Softmax maps output into probabilities!



$$\varphi^{(o)}(.) = \text{Softmax}(.) \qquad \varphi^{(h)}(.) = \sigma(.) \text{ (Sigmoid or ReLU)}$$

$$a_c^{(o)} = \frac{e^{z_c^{(o)}}}{\sum_{k=1}^{C} e^{z_k^{(o)}}}$$

input

all raw outputs

$$L(y, z^{(o)}) = -\sum_{c=1}^{C} y_c \log\left(\frac{e^{z_c^{(o)}}}{\sum_{k=1}^{C} e^{z_k^{(o)}}}\right)$$

$$\frac{\partial L}{\partial z_c^{(o)}} = a_c^{(o)} - y_c \qquad \boxed{\frac{\partial L}{\partial w_{j,c}^{(o)}} = \frac{\partial L}{\partial z_c^{(o)}} \cdot a_j^{(h)} \quad \frac{\partial L}{\partial b_c^{(o)}} = \frac{\partial L}{\partial z_c^{(o)}}}$$

$$\frac{\partial L}{\partial a_j^{(h)}} = \sum_c \left(\frac{\partial L}{\partial z_c^{(o)}} \cdot \frac{\partial z_c^{(o)}}{\partial a_j^{(h)}}\right)$$

$$\frac{\partial L}{\partial z_j^{(h)}} = \frac{\partial L}{\partial a_j^{(h)}} \cdot \frac{\partial a_j^{(h)}}{\partial z_j^{(h)}} \qquad \boxed{\frac{\partial L}{\partial w_{i,j}^{(h)}} = \frac{\partial L}{\partial z_j^{(h)}} \cdot x_i \quad \frac{\partial L}{\partial b_j^{(h)}} = \frac{\partial L}{\partial z_j^{(h)}}}$$

Regularization

- Need to talk about **Bias & Variance**

acceptable err

High Variance	High Bias
• Training error is much **lower** than **validation error**.	• Training err **higher than ε**
~ We have to **reduce model complexity**.	~ Use **more complex model**
~ We can do **bagging, under-training, add training data**.	~ Add **new features, boosting**

- We can apply a variety of regulation methods in the learning of ANN.
- Play a very important role specially when training set is small.
  - Weight decay
  - Early stopping
  - Dropout
  - Data Augmentation

Weight Decay

- Add L^p norm-regulazion terms to the objective function
- With L^2 norm: informally called weight decay because:
  - tries to push magnitude of W towards zero!

Without regulation

Loss function: $Q(W)$

Gradient Descent update: $W := W - \alpha \frac{\partial Q(W)}{\partial W}$

$\longrightarrow$

With L^2 regulation

Loss Function: $Q(W) + \frac{\lambda}{2} \cdot \|W\|_2$

GD update: $W := W - \alpha \frac{\partial Q(W)}{\partial W} - \lambda W$

- There is no regulation for biases:
  - they don't change the shape!

Dropout Regularization

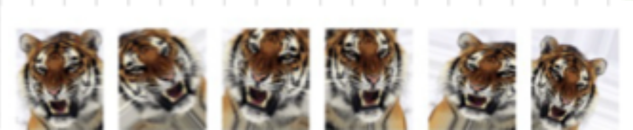
- Injecting noise in learning process
- $\forall$ training sample (in mini batch):
  - make a thinned network by randomly dropping out nodes
  - Forward & back-propagation are done on thinned network
- Gradients for updating each weight are averaged over the training samples.
  - the dropped nodes report a gradient = 0 for their weights.
- Each node has the chance to be dropped every time, forced to redundant representation
  - this prevents overfitting.
- Less complex methods → avoid noise → prevent overfitting
- The concept is that full network learns on scaled-down version of itself.
- For prediction the full network is used.

Early Stopping

- When performance on validation set is getting worse → Stop!

Data Augmentation

- Generating more alternative training samples from the raw training data.
  - Inject small noise into raw data — altering



GD with momentum

- Add a history of the gradients
- Give more weight to more recent gradients

$$V_{t+1} = \beta V_t + \alpha \nabla L(W_t)$$

$$W_{t+1} = W_t - V_{t+1}$$

- Gets momentum & accelerates if the gradient was in the same direction
- Decreases if direction was made with large steps.