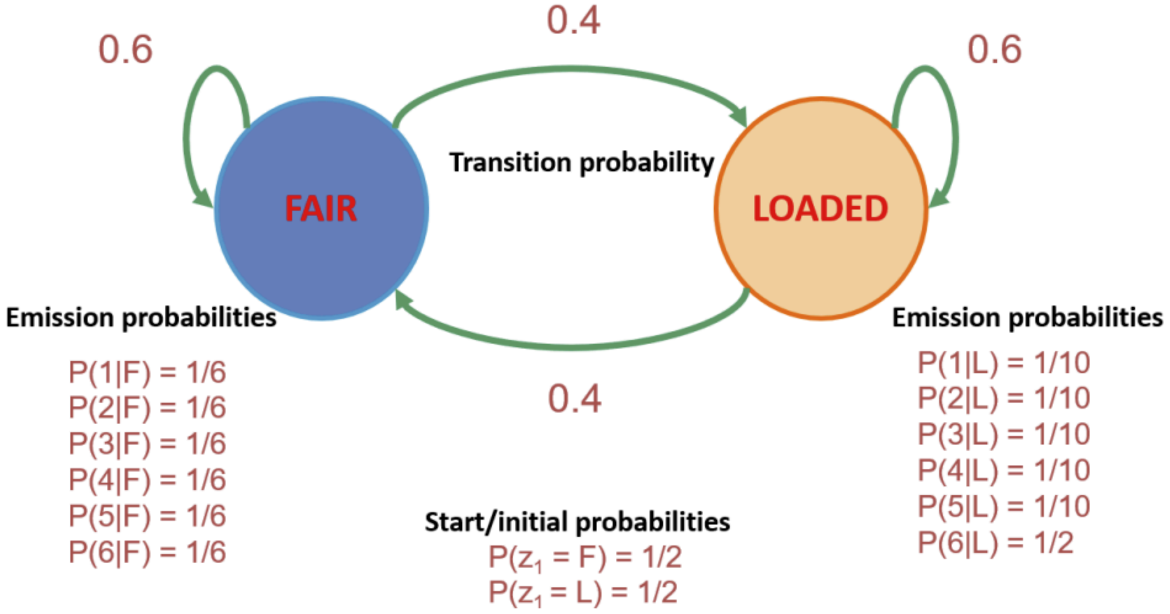
1 Calculate Joint Probabilities in HMM



Emission probabilities

P(1|F) = 1/6
P(2|F) = 1/6
P(3|F) = 1/6
P(4|F) = 1/6
P(5|F) = 1/6
P(6|F) = 1/6

Emission probabilities

P(1|L) = 1/10
P(2|L) = 1/10
P(3|L) = 1/10
P(4|L) = 1/10
P(5|L) = 1/10
P(6|L) = 1/2

Start/initial probabilities

$P(z_1 = F) = 1/2$
$P(z_1 = L) = 1/2$

x = 6, 3, 1, 2, 4

z = Load, Fair, Fair, Load, Load

$$P(x,z) = P(6|L)\,P(3|F)\,P(1|F)\,P(2|L)\,P(4|L)$$

$$\times \frac{1}{2}\,P(F|L)\,P(F|F)\,P(L|F)\,P(L|L)$$

$$P(x,z) = \frac{1}{2} \times \frac{1}{6} \times \frac{1}{6} \times \frac{1}{10} \times \frac{1}{10} \times$$

$$\frac{1}{2} \times 0.4 \times 0.6 \times 0.4 \times 0.6$$

$$= \frac{1}{250000} = 4 \times 10^{-6}$$

2 Viterbi algorithm

The V Matrix

	t = 1	t = 2	t = 3
k = 1 (Fair)	$\frac{1}{6} * \frac{1}{2} = \frac{1}{12}$	$\frac{1}{6} \max\left(0.6 \times \frac{1}{12}, 0.4 \times \frac{1}{20}\right)$ $= \frac{1}{120}$	$\frac{1}{6} \max\left(0.6 \times \frac{1}{120}, 0.4 \times \frac{1}{60}\right)$ $= \frac{1}{900}$
k = 2 (Loaded)	$\frac{1}{10} * \frac{1}{2} = \frac{1}{20}$	$\frac{1}{2} \max\left(0.4 \times \frac{1}{12}, 0.6 \times \frac{1}{20}\right)$ $= \frac{1}{60}$	$\frac{1}{10} \max\left(0.4 \times \frac{1}{120}, 0.6 \times \frac{1}{60}\right)$ $= \frac{1}{1000}$
observation	"4"	"6"	"3"

The Ptr Matrix

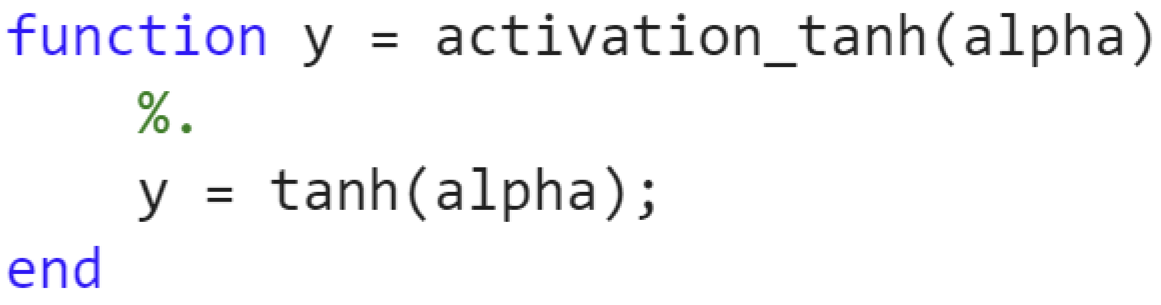
	t = 1	t = 2	t = 3
k = 1 (Fair)	0	$\arg\max\left(0.6 \times \frac{1}{12}, 0.4 \times \frac{1}{20}\right)$ $\arg\max\left(\frac{1}{20}, \frac{1}{50}\right) = 1$	$\arg\max\left(0.6 \times \frac{1}{120}, 0.4 \times \frac{1}{60}\right)$ $\arg\max\left(\frac{1}{200}, \frac{1}{150}\right) = 2$
k = 2 (Loaded)	0	$\arg\max\left(0.4 \times \frac{1}{12}, 0.6 \times \frac{1}{20}\right)$ $\arg\max\left(\frac{1}{30}, \frac{3}{100}\right) = 1$	$\arg\max\left(0.4 \times \frac{1}{120}, 0.6 \times \frac{1}{60}\right)$ $\arg\max\left(\frac{1}{300}, \frac{1}{100}\right) = 2$
observation	"4"	"6"	"3"

$$P(x, z^*) = \max_k V_k(n) = \frac{1}{900}$$

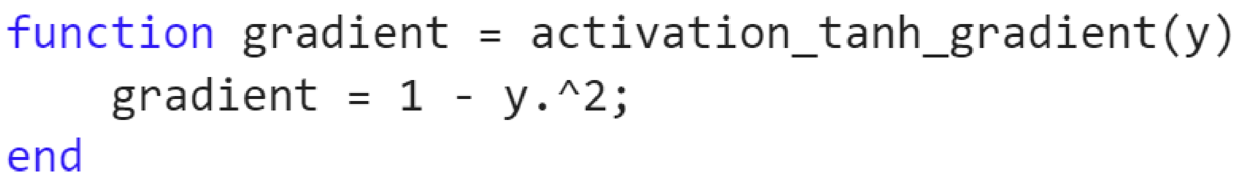
$$z^* = \{fair, loaded, fair\}$$

3 Feed Forward Neural Networks

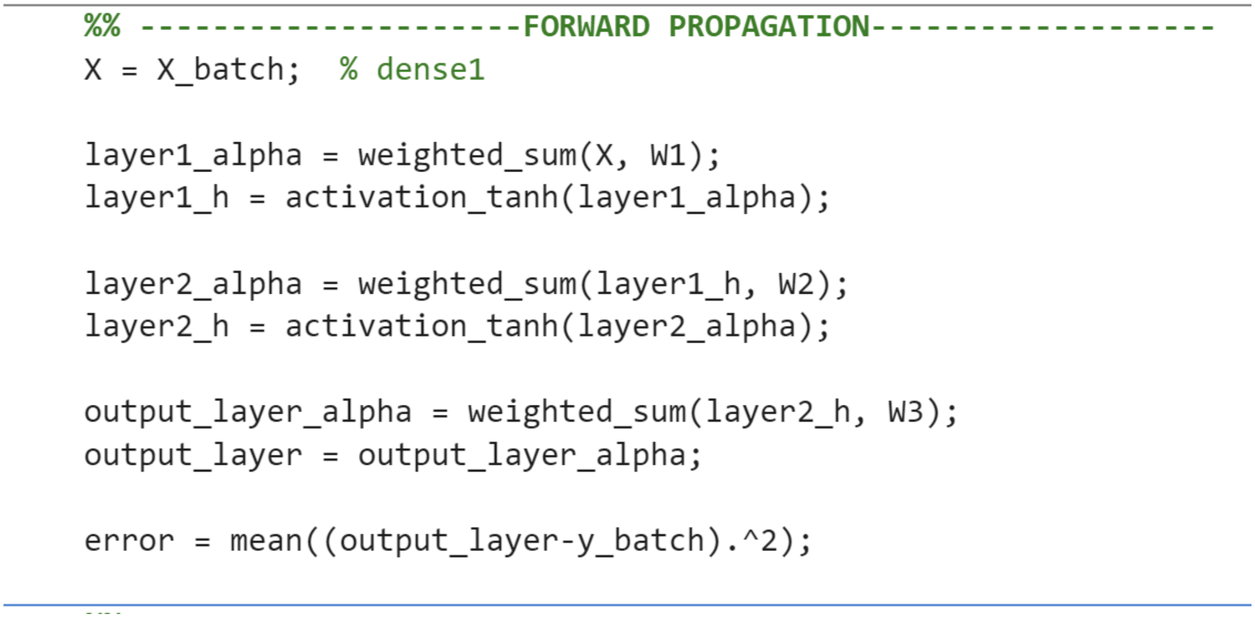
These were the lines of code that were changed



```
function y = activation_tanh(alpha)
    %.
    y = tanh(alpha);
end
```



```
function gradient = activation_tanh_gradient(y)
    gradient = 1 - y.^2;
end
```



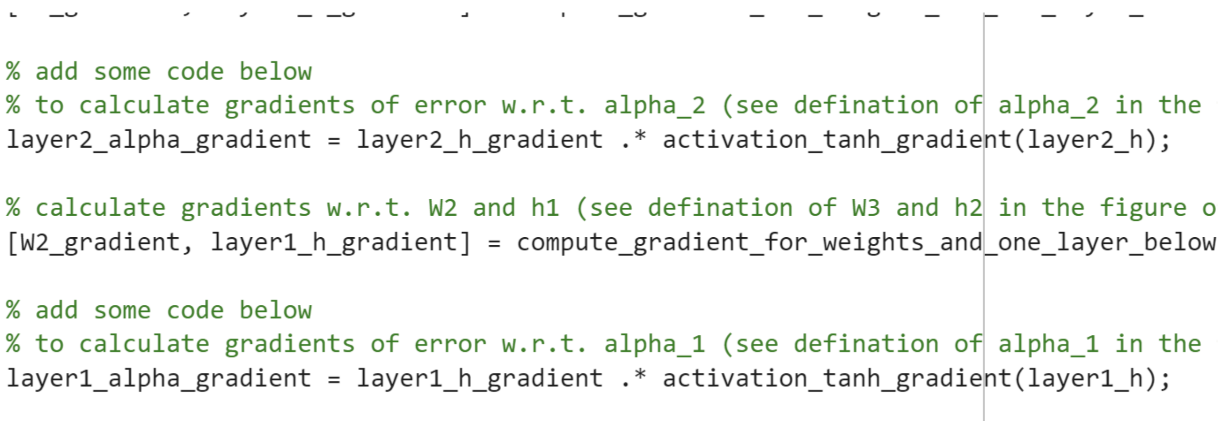
```
    %% --------------------FORWARD PROPAGATION------------------
    X = X_batch;  % dense1

    layer1_alpha = weighted_sum(X, W1);
    layer1_h = activation_tanh(layer1_alpha);

    layer2_alpha = weighted_sum(layer1_h, W2);
    layer2_h = activation_tanh(layer2_alpha);

    output_layer_alpha = weighted_sum(layer2_h, W3);
    output_layer = output_layer_alpha;

    error = mean((output_layer-y_batch).^2);
```



```
% add some code below
% to calculate gradients of error w.r.t. alpha_2 (see defination of alpha_2 in the
layer2_alpha_gradient = layer2_h_gradient .* activation_tanh_gradient(layer2_h);

% calculate gradients w.r.t. W2 and h1 (see defination of W3 and h2 in the figure o
[W2_gradient, layer1_h_gradient] = compute_gradient_for_weights_and_one_layer_below

% add some code below
% to calculate gradients of error w.r.t. alpha_1 (see defination of alpha_1 in the
layer1_alpha_gradient = layer1_h_gradient .* activation_tanh_gradient(layer1_h);
```