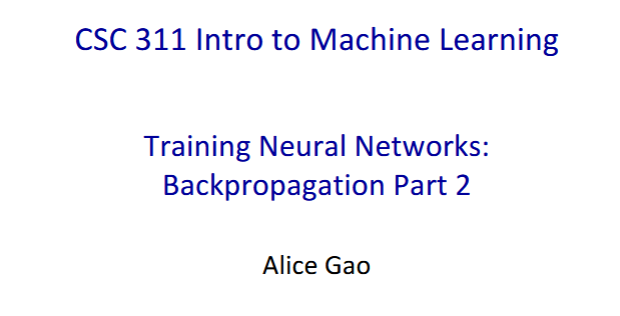
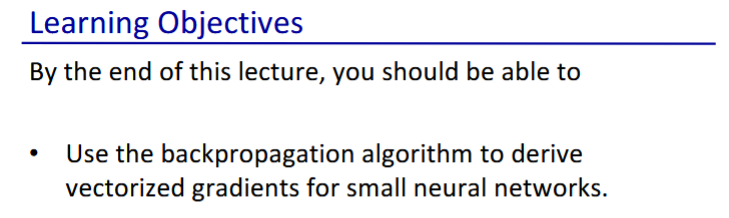
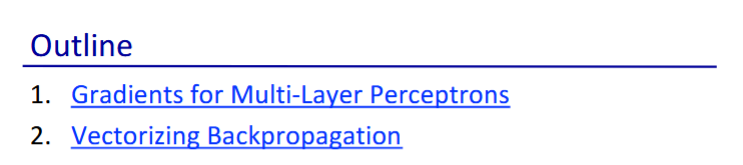
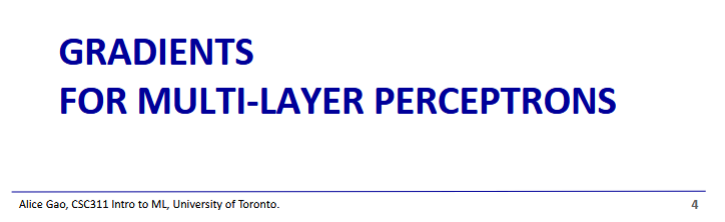
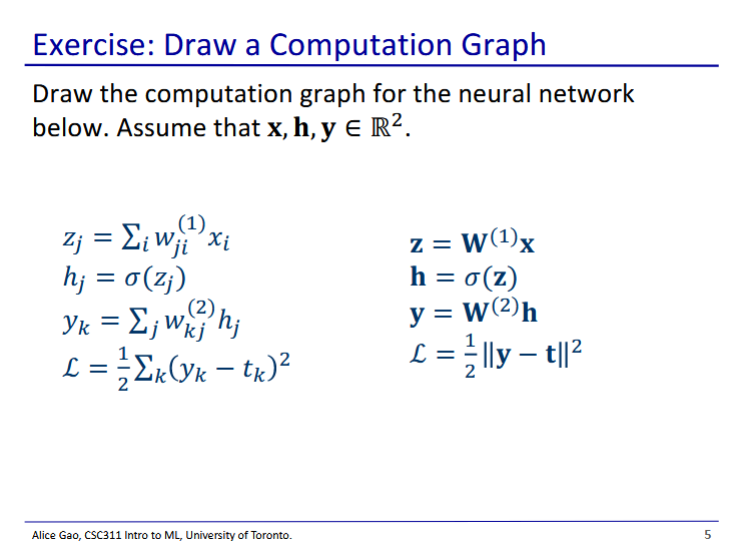
| **Test stuff**   * Average is around 60%, Prof. will adjust it to ~70% * Grades getting released later today on Crowdmark |
| --- |



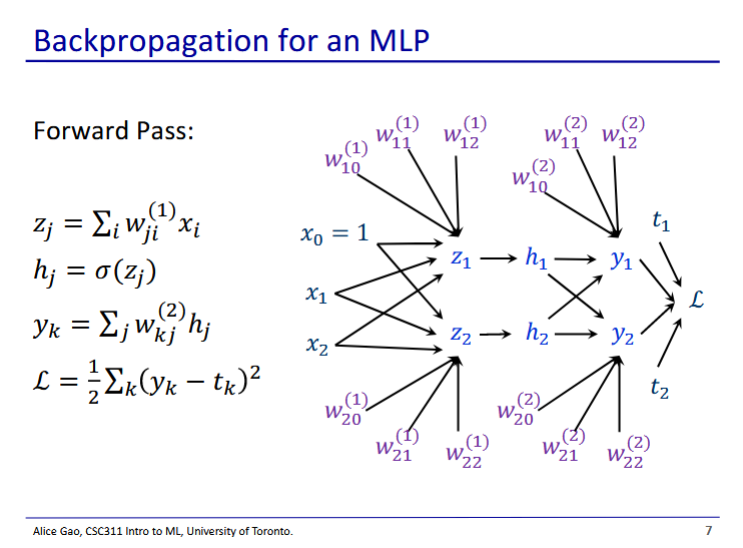




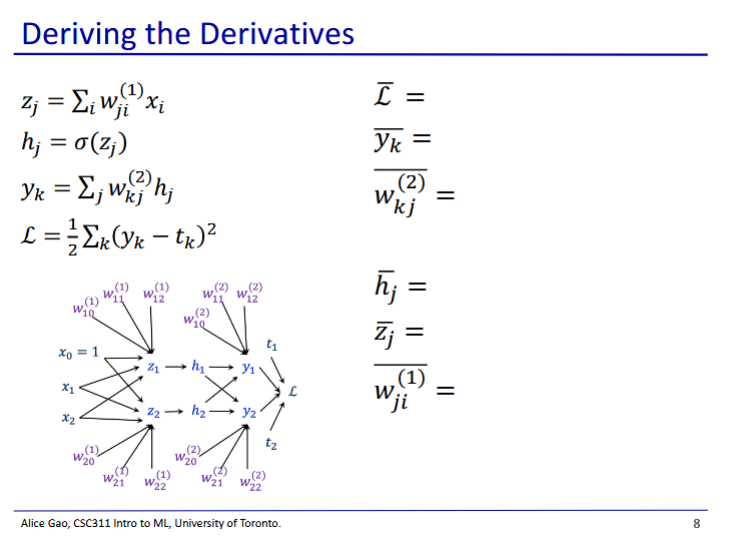




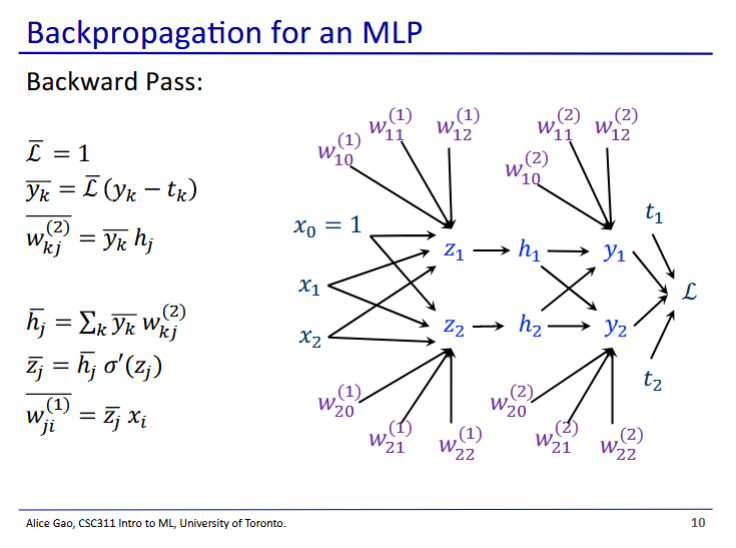
* Left shows non-vectorised equations, right shows vectorised equations
* How many layers in this neural network?
  + 2 layers
  + X is input layer
  + First layer is h
    - z is linear combination of inputs, z is then passed through the activation function to get h
  + Second layer is y
    - Has no activation function, just weights applied to h
* How many nodes in each layer?
  + 2, since



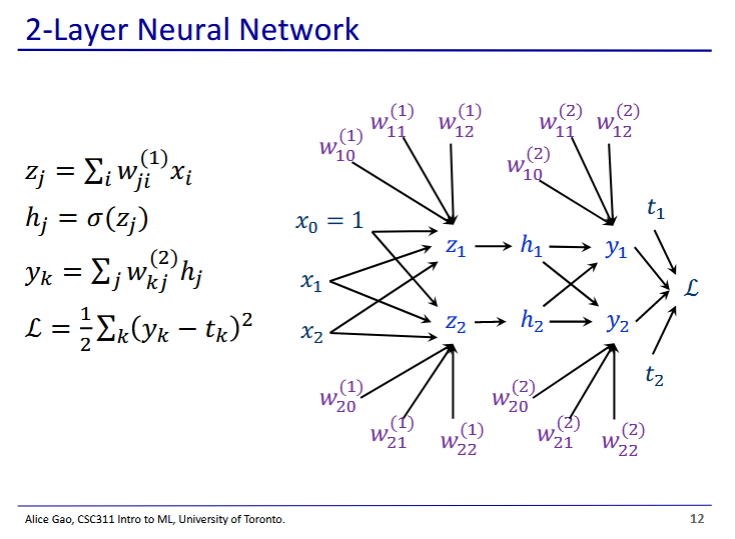
* This is the computation graph
* Note: dummy feature was forgotten for the h layer

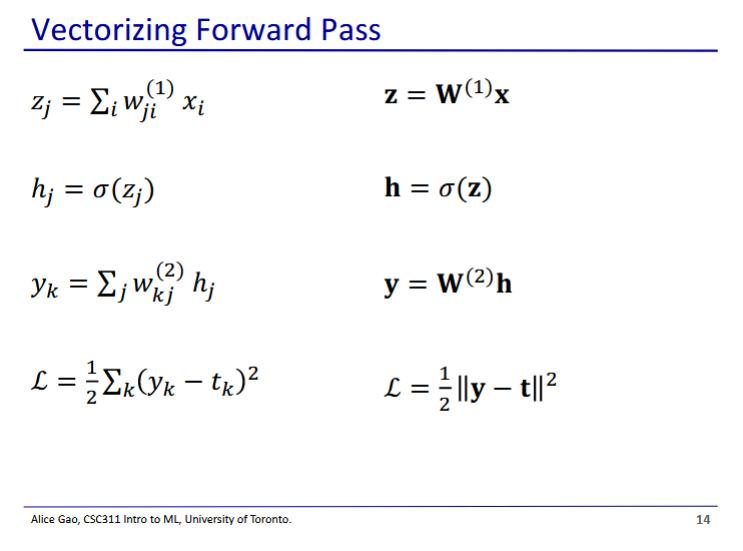


* + For this one we use the multivariate chain rule, as there are 2 paths from hj to L
  + is the derivative of the activation function

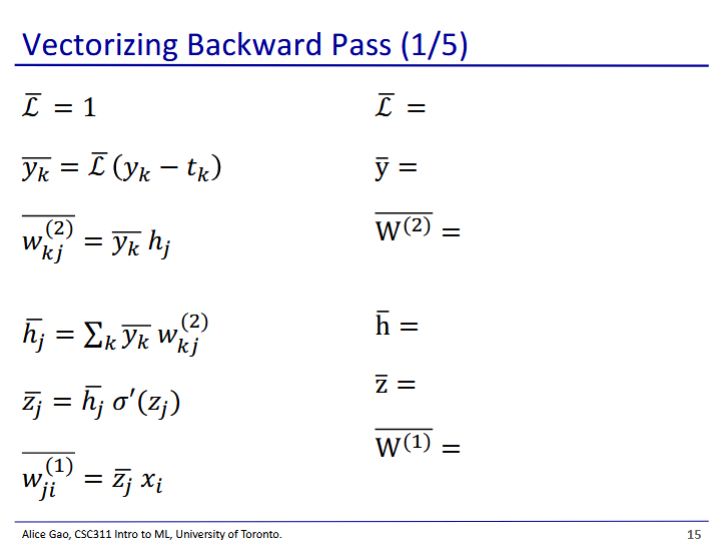




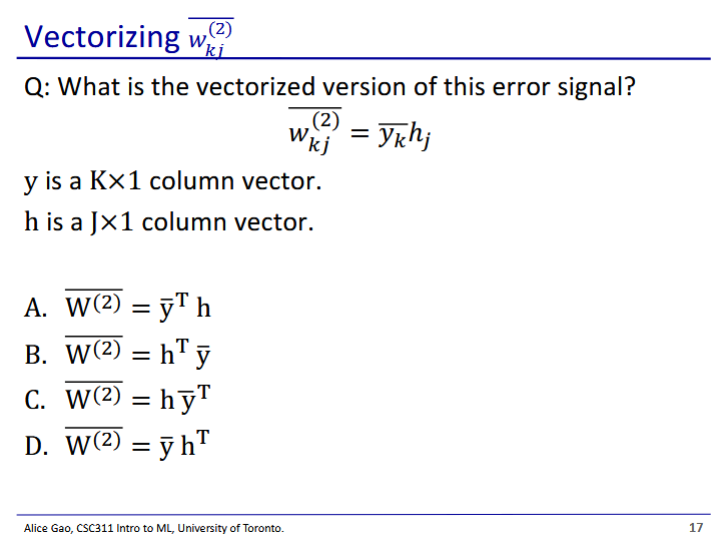




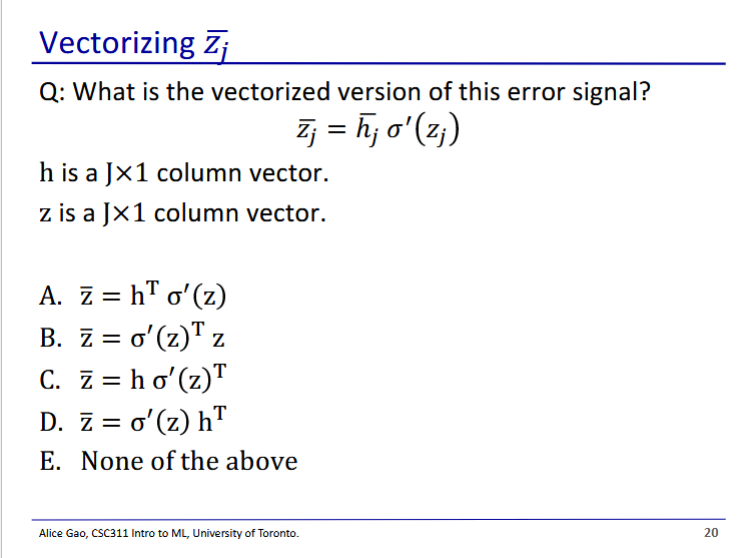
* We also use for the logistic activation function for vectors, where it applies to each component of vector **z**
  + Gives us a resulting vector **h**



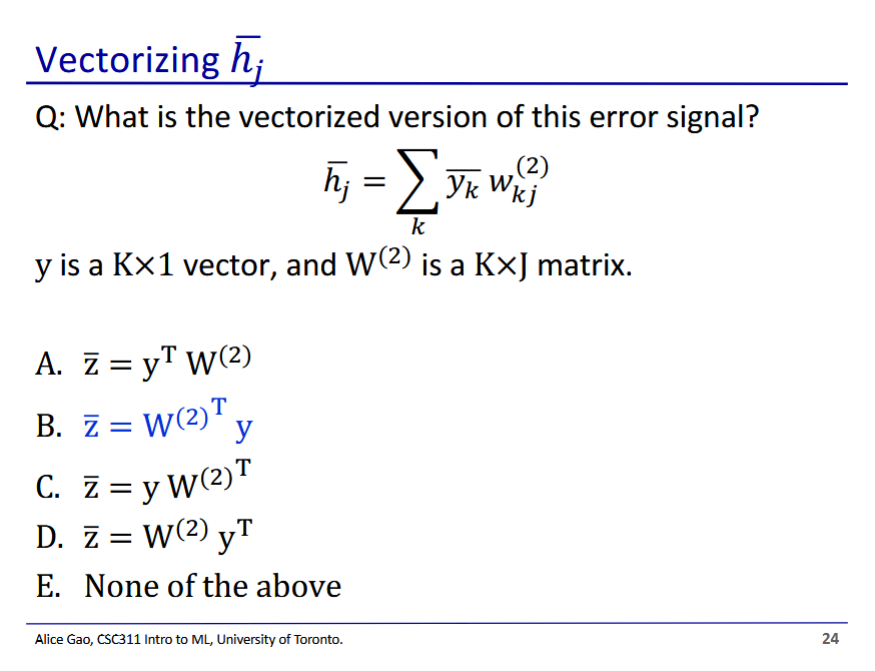
* Filled out version at end



* Answer: D
  + This is the outer product operation, which produces a matrix
  + y is a (k by 1) vector, is a (1 by j) vector, produces a k by j matrix



* Answer: None of the above
* We want a jx1 vector
* We get this using a Hadamard product
  + This is multiplication of each component pair



* y should have a line over it ()
* We want a vector of size (j \* 1),W(2) has dimensions (k \* j), and has dimensions of (k \* 1)

